Exploring the slowness principle in the auditory domain

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

Doctor rerum naturalium
(Dr. rer. nat.)
im Fach Biophysik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät I
Humboldt-Universität zu Berlin

von
Herr Dipl.-Phys. Tiziano Zito

Präsident der Humboldt-Universität zu Berlin:
Prof. Dr. Jan-Hendrik Olbertz

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät I:
Prof. Dr. Andreas Herrmann

Gutachter:
1. Prof. Dr. Laurenz Wiskott
2. Prof. Dr. Felix Wichmann
3. Prof. Dr. Richard Kempter

eingereicht am: 28.10.2010
Tag der mündlichen Prüfung: 28.11.2011

Si sta come
d'autunno
sugli alberi
le foglie

Abstract

In this thesis we develop models and algorithms based on the slowness principle in the auditory domain. Several experimental results as well as the successful results in the visual domain indicate that, despite the different nature of the sensory signals, the slowness principle may play an important role in the auditory domain as well, if not in the cortex as a whole. Different modeling approaches have been used, which make use of several alternative representations of the auditory stimuli. We show the limitations of these approaches. In the domain of signal processing, the slowness principle and its straightforward implementation, the Slow Feature Analysis algorithm, has been proven to be useful beyond biologically inspired modeling. A novel algorithm for nonlinear blind source separation is described that is based on a combination of the slowness and the statistical independence principles, and is evaluated on artificial and real-world audio signals. The Modular toolkit for Data Processing open source software library is additionally presented.

Keywords: temporal slowness, slow feature analysis, nonlinear blind source separation, independent component analysis, auditory system

Zusammenfassung

In dieser Arbeit werden - basierend auf dem Langsamkeitsprinzip - Modelle und Algorithmen für das auditorische System entwickelt. Verschiedene experimentelle Ergebnisse, sowie die erfolgreichen Ergebnisse im visuellen System legen nahe, dass, trotz der unterschiedlichen Beschaffenheit visueller und auditorischer sensorischer Signale, das Langsamkeitsprinzip auch im auditorischen System eine bedeutsame Rolle spielen könnte, und vielleicht auch im Kortex im Allgemeinen. Es wurden verschiedene Modelle für unterschiedliche Repräsentationen des auditorischen Inputs realisiert. Es werden die Beschränkungen der jeweiligen Ansätze aufgezeigt. Im Bereich der Signalverarbeitung haben sich das Langsamkeitsprinzip und dessen direkte Implementierung als Signalverarbeitungsalgorithmus, Slow Feature Analysis, über die biologisch inspirierte Modellierung hinaus als nützlich erwiesen. Es wird ein neuer Algorithmus für das Problem der nichtlinearen blinden Signalquellentrennung beschrieben, der auf einer Kombination von Langsamkeitsprinzip und dem Prinzip der statistischen Unabhängigkeit basiert, und der anhand von künstlichen und realistischen Audiosignalen getestet wird. Außerdem wird die Open Source Software Bibliothek Modular toolkit for Data Processing vorgestellt.

Schlagwörter: zeitliche Langsamkeit, langsame Komponenten Analyse, nichtlineare blinde Signalquellentrennung, unabhängige Komponenten Analyse, auditorisches System

Kurzfassung in deutscher Sprache

Wir erleben unsere Umwelt als relativ stabil und über die Zeit konstant. Die sensorischen Signale, die von den Sinneszellen registriert werden, variieren jedoch auf einer anderen Zeitskala als die für unsere Wahrnehmung relevanten Umwelteigenschaften, wie z.B. die Identität und Position von visuellen Objekten oder die Identität und Position eines Sprechers. Die Lichtintensität, mit der ein retinaler Photorezeptor erregt wird oder der Schalldruck, mit der die tympanische Membran (Trommelfell) angeregt wird, variieren um Größenordnungen schneller als die dazugehörigen wahrnehmungsrelevanten Eigenschaften der Umwelt.

Das Prinzip der *zeitlichen Langsamkeit*, oder einfach *Langsamkeit*, besagt, dass man eine stabile Repräsentation der Umwelt erhalten könnte, indem man aus dem schnell variierenden sensorischen Signal die langsam variierenden Signalanteile extrahiert. Auf diese Weise würde eine Repräsentation generiert, die gegenüber typischen Transformationen des sensorischen Signals - wie Rotation und Translation im visuellen System oder Verstärkung und Phasenverschiebung im auditorischen System - invariant ist. Die Anwendbarkeit des Langsamkeitsprinzip für die Organisationsweise des Gehirns wurde in mehreren Studien vorgeschlagen (siehe Földiák, 1991; Mitchison, 1991; Stone and Bray, 1995; Stone, 1996; Wiskott, 1998; Wiskott and Sejnowski, 2002; Berkes and Wiskott, 2005). Berkes and Wiskott (2005) konnten zeigen, dass ein einfaches unüberwachtes Modell des primären visuellen Kortex, das auf dem Langsamkeitsprinzip beruht, Mechanismen hervorbringt, deren Eigenschaften den komplexen Zellen im primären Kortex ähnelten. Verschiedene Arbeiten (Franzius, Sprekeler, and Wiskott, 2007b; Sprekeler, Michaelis, and Wiskott, 2007; Dähne, Wilbert, and Wiskott, 2009; Hinze, Wilbert, and Wiskott, 2009) bestätigten später die Anwendbarkeit des Langsamkeitsprinzips als mächtiges und gleichzeitig biologisch plausibles computationales Prinzip der Selbstorganisation des visuellen Kortex.

In dieser Arbeit werden - basierend auf dem Langsamkeitsprinzip - Modelle und Algorithmen für das auditorische System entwickelt. Zunächst muss man sich bewusst machen, dass auditorische und visuelle Signale sehr verschieden sind. Die Reichhaltigkeit eines Klanges wird in zwei (beide Ohren) eindimensionale Zeitreihen komprimiert, die einen dynamischen Bereich von 13 Zehnerpotenzen physikalischer Amplitudenvariationen umspannen und eine zeitliche Auflösung von mehr als 40kHz haben. Mit 130 Millionen retinalen Photorezeptoren sind visuelle Signale extrem hoch-dimensional, umspannen einen dynamischen Bereich von 7-10 Zehnerpotenzen (Ferwerda, Pattanaik, Shirley, and Greenberg, 1996), variieren aber nur auf einer Zeitskala von 50Hz. Den-

noch sind sich beide sensorischen Kortexareale sehr ähnlich, sowohl anatomisch als auch funktionell, wie in mehreren spektakulären "rewiring" Experimenten gezeigt wurde (Roe, Pallas, Kwon, and Sur, 1992; Sur, Angelucci, and Sharma, 1999; Sharma, Angelucci, and Sur, 2000; von Melchner, Pallas, and Sur, 2000). Wenn mittels neonataler Chirurgie bei Frettchen retinale Signale in auditorische Pfade umgelenkt wurden, dann entwickelten sich im auditorischen Kortex dieser "rewired" Tiere Neurone, die dieselben Eigenschaften hatten wie Neurone im primären visuellen Kortex normaler Tiere. Derartige Beobachtungen sowie die erfolgreichen Ergebnisse im visuellen System legen nahe, dass, trotz der unterschiedlichen sensorischen Signale, das Langsamkeitsprinzip auch im auditorischen System eine bedeutsame Rolle spielen könnte, und vielleicht auch im Kortex im Allgemeinen.

In Kapitel 5 wird im Detail beschrieben und diskutiert wie hier versucht wurde, Modelle des auditorischen Kortex basierend auf dem Langsamkeitsprinzip zu entwickeln. Es wurden verschiedene Modelle für unterschiedliche Repräsentationen des auditorischen Inputs realisiert. Es werden die Beschränkungen der jeweiligen Ansätze aufgezeigt. Die Beschaffenheit auditorischer Signale, insbesondere ihre Andersartigkeit als visuelle Signale erklärt die beobachteten Ergebnisse. Bis auf einfache Fälle sind die Ergebnisse nicht zufriedenstellend. Die Gründe dafür, warum das Langsamkeitsprinzip im auditorischen System zu unbefriedigenden Ergebnissen führt, sind nicht vollständig verstanden. Einige mögliche Erklärungen werden am Ende des Kapitels diskutiert.

Das Langsamkeitsprinzip und dessen direkte Implementierung als Signalverarbeitungsalgorithmus, Slow Feature Analysis (SFA), haben sich aber über die biologisch inspirierte Modellierung hinaus als nützlich erwiesen. Im Bereich der Signalverarbeitung hat sich SFA als mächtiges Werkzeug zur Problemanalyse etabliert. So wurde beispielsweise gezeigt, das lineare SFA zu äquivalenten Ergebnissen führt wie die zeitkorrelationsbasierte Unabhängige Komponenten-Analyse (Independent Component Analysis, ICA) (Blaschke, Berkes, and Wiskott, 2006). SFA wurde auch für die Erkennung handgeschriebener Zahlen benutzt (Berkes, 2005). In den Kapiteln 2 und 3 geht es um das Problem der nichtlinearen blinden Signalquellentrennung (Blind Source Separation). Es wird ein neuer Algorithmus beschrieben, Independent Slow Feature Analysis (ISFA), der auf einer Kombination von Langsamkeitsprinzip und dem Prinzip der statistischen Unabhängigkeit basiert und der anhand von künstlichen und realistischen Audiosignalen getestet wird. Für viele Fälle liefert der Algorithmus vernünftige Ergebnisse. Eine Analyse der Fälle, in denen er nicht funktioniert, zeigt, dass das Prinzip der statistischen Unabhängigkeit, zumindest wenn es nur auf der Basis von Statistiken zweiter Ordnung beruht, nicht ausreicht, um zufriedenstellende Performanz zu gewährleisten.

In Kapitel 4 wird das *Slowness* Theorem vorgestellt, das die formale Grundlage für ISFA bildet.

Alle in dieser Arbeit berichteten Ergebnisse wurden mittels computerbasierter Simulationen und Analysen gewonnen. Ich bin der Überzeugung, dass es notwendig ist, den Programmcode, der die Ergebnisse hervorgebracht hat zu veröffentlichen, um die Er-

gebnisse für andere computationale Neurowissenschaftler nutzbar zu machen. Deshalb haben Dr. Pietro Berkes und ich entschieden, den Programmcode in einer Bibliothek als Open Source Software zu organisieren, die gut dokumentiert und frei verfügbar ist. Diese anfängliche Anstrengung ist in ein community-driven Signalverarbeitungs-Projekt gemündet, das bereits jetzt eine der meist genutzten Bibliotheken innerhalb der Python-basierten machine-learning community ist. Die Bibliothek wird in Kapitel 6 vorgestellt.

Acknowledgement

The work presented in this thesis has been accomplished at the Institute for Theoretical Biology of the Humboldt University in Berlin, Germany, in the group lead by Prof. Laurenz Wiskott. Laurenz has been a reliable and omiscient guide in the years of my doctoral studies. Our everlasting discussions and his amazing intuition have shaped my idea of what science is all about. My friends in Laurenz's group have never lost hope about this thesis to be finally written, thanks to Pietro Berkes, Tobias Blaschke, Irina Erchova, Mathias Franzius, Henning Sprekeler, and Niko Wilbert.

Contents

1 Introduction

We perceive the external world as a relatively stable and consistent system. The sensory signals that our receptors measure, however, vary on a very different time scale than the relevant properties of the environment, like for example identity and position of objects in the visual domain, or identity and position of speakers in the auditory domain. The intensity of the light hitting a retinal photo-receptor or the intensity of the air pressure wave hitting the tympanic membrane show a fine structure that varies on a time scale several orders of magnitude faster than the behaviorally relevant features of the environment.

The principle of *temporal slowness*, or just *slowness*, states that if we extract slowly varying signals from the quickly varying sensory inputs we may obtain a representation of the environment that is stable. We generate a representation that is invariant under typical transformations of the sensory input, like rotation and translation in the visual domain or amplification and time shift in the auditory domain. Slowness has been proposed as a principle for the cerebral cortex in several studies (see Földiák, 1991; Mitchison, 1991; Stone and Bray, 1995; Stone, 1996; Wiskott, 1998; Wiskott and Sejnowski, 2002; Berkes and Wiskott, 2005). In particular Berkes and Wiskott (2005) show how a simple unsupervised model of the primary visual cortex based on the slowness principle yields a set of units with properties closely resembling those of complex cells in the primary visual cortex. Several successive studies (Franzius et al., 2007b; Sprekeler et al., 2007; Dähne et al., 2009; Hinze et al., 2009) corroborate these early results by confirming slowness as a powerful and at the same time biologically plausible computational principle for the self-organization of the visual cortex.

In this thesis we develop models and algorithms based on the slowness principle in the auditory domain. Audio signals have a completely different nature compared to visual signals. The richness of the sound landscape is compressed in two one-dimensional time-series spanning 13 order of magnitudes of dynamic range and a temporal resolution higher than 40 kHz. With 130 million retinal photoreceptors, visual signals are extremely high-dimensional, span 7 to 10 orders of magnitude of dynamic range (Ferwerda et al., 1996), but vary on a much slower time scale of the order of 50 Hz. Nonetheless the cortex shows a high degree of homogeneity, both in the anatomical structure as well as in the functional organization. This has been unquestionably shown in several spectacular "rewiring" experiments (Roe et al., 1992; Sur et al., 1999; Sharma et al., 2000; von Melchner et al., 2000). When retinal input is routed to the auditory pathway in ferrets by means of neonatal surgical manipulations, neurons in the primary auditory cortex of

"rewired" animals show the same properties of neurons of the primary visual cortex in normal animals. These observations together with the successful results in the visual domain indicate that, despite the different nature of the sensory signals, the slowness principle may play an important role in the auditory domain as well, if not in the cortex as a whole. In Chapter 5 an attempt to develop models of the auditory cortex based on the slowness principle is presented and discussed in detail.

The slowness principle and its straightforward implementation as a signal processing algorithm, namely the Slow Feature Analysis (SFA) algorithm (Wiskott and Sejnowski, 2002) has been proven to be useful beyond biologically inspired modeling. In the domain of signal processing, SFA is a powerful tool to solve several problems. For example, linear SFA has been shown to be equivalent to time-correlation based Independent Component Analysis (ICA) (Blaschke et al., 2006). Furthermore, SFA has been used to perform hand-written digit recognition by Berkes (2005). Chapters 2 and 3 deal with the problem of nonlinear blind source separation. A novel algorithm, Independent Slow Feature Analysis (ISFA), is described that is based on a combination of the slowness and the statistical independence principles, and is evaluated on artificial and real-world audio signals. Chapter 4 presents the *Slowness* theorem, which establishes the formal foundation of the ISFA algorithm.

All of the results presented in this thesis have been obtained by means of computer-assisted simulations and analysis. I am convinced that a necessary step in making those results useful for the computational neuroscience community is to publish the code that yielded them. Together with Dr. Pietro Berkes we decided to organize this code in an open source library, freely available and well documented. This initial effort has grown into a community-driven signal processing framework, which is already one of the most used libraries in the Python machine learning community. This library is presented in Chapter 6.

2 Blind Source Separation, Independent Component Analysis, and Slow Feature Analysis

The problem of Blind Source Separation (BSS) has become an established subject of research in the signal processing domain since the early works by Jutten and Herault (1985) and Comon, Jutten, and Herault (1991). Given a number of statistically independent unkown signals arbitrarily composed to form an observed mixture, the problem consists in retrieving them based solely on the assumption that the signals are independent of each other. Because no prior knwowledge is assumed about the signal or the mixing system the problem is said to be *blind*. The problem of BSS is of interest in the most diverse domains of research: telecommunication, speech recognition, computer vision, brain imaging and brain modeling (a comprehensive list of references can be found, for example, in Comon and Jutten, 2010).

Let $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]^{\mathrm{T}}$ be the observed mixture of the unkown independent source signals $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]^{\mathrm{T}}$, where for simplicity we assume the number of observed signals to be equal to that of the unkown sources. The mixing system is a mapping of $\mathbf{s}$ onto $\mathbf{x}$, namely:

$$\mathbf{x}(t) = \mathcal{F}(\mathbf{s}(t)). \tag{2.1}$$

In general $\mathcal{F}$ can be nonlinear and time-dependent. The goal of BSS is to find a separating system $\mathcal{G}$, such that the output vector $\mathbf{u}(t)$ obtained by

$$\mathbf{u}(t) = \mathcal{G}(\mathbf{x}(t)) \tag{2.2}$$

has components $u_i(t)$ which are dependent only on a single component $s_j(t)$ of the source signal:

$$u_i = h_i(s_{\sigma(i)}(t)), \quad i = 1, \ldots, N \tag{2.3}$$

where $\sigma(i)$ is a permutation of the indices $i = 1, \ldots, N$. The mapping h can be nonlinear in general, but it must be invertible, so that all information present in the original source signal component s_j is retained in the corresponding output signal component h_i. If the mixing system is linear, than h_i reduces to a linear mapping.

2.1 Linear BSS and ICA

Let $\mathbf{x}(t) = [x_1(t), \ldots, x_N(t)]^{\mathrm{T}}$ be a linear mixture of the source signals $\mathbf{s}(t) = [s_1(t), \ldots, s_N(t)]^{\mathrm{T}}$ and be defined by

$$\mathbf{x}(t) = \mathbf{A}\mathbf{s}(t)\,, \tag{2.4}$$

with an invertible $N \times N$ mixing matrix $\mathbf{A}$. The goal of linear BSS is then to recover the unknown source signal $\mathbf{s}(t)$ from the observable $\mathbf{x}(t)$ without any prior information. The only assumption is that the source signal components are statistically independent. Given only the observed signal $\mathbf{x}(t)$ we want to find a matrix $\mathbf{R}$ such that the components of

$$\mathbf{u}(t) = \mathbf{Q}\mathbf{y}(t) = \mathbf{Q}\mathbf{W}\mathbf{x}(t) = \mathbf{R}\mathbf{x}(t)\,, \tag{2.5}$$

are mutually statistically independent. Here we have divided $\mathbf{R}$ into two parts. First a *whitening* transformation $\mathbf{y}(t) = \mathbf{W}\mathbf{x}(t)$ with whitening matrix $\mathbf{W}$ is applied, resulting in uncorrelated signal components $y_i(t)$ with unit variance and zero mean, where we have assumed $\mathbf{x}(t)$ and also $\mathbf{s}(t)$ to have zero mean. In a second step a transformation $\mathbf{u}(t) = \mathbf{Q}\mathbf{y}(t)$ with orthogonal $\mathbf{Q}$ (Comon, 1994) results in statistically independent components $u_i(t)$.

The method of finding a representation of the observed data such that the components are mutually statistically independent is called Independent Component Analysis (ICA). It has been proven that ICA solves the linear BSS problem, apart from the fact that the source signal components can only be recovered up to scaling and permutation (Comon, 1994).

There exists a variety of algorithms performing linear ICA and therefore linear BSS. They can be divided into two classes (Cardoso, 2001):

(i) independence is achieved by optimizing a criterion that requires higher order statistics (*high-order ICA*)

(ii) the optimization criterion requires auto-correlations or non-stationarity of the source signal components (*second-order ICA*).

Algorithms of class (i) consider signals to be independent if their joint distribution is equal to the product of their marginal distributions (e.g. Cardoso and Souloumiac, 1993; Hyvärinen, 1999; Lee, Girolami, and Sejnowski, 1999). A suitable measure for independence is then the Kullback-Leibler divergence, which measures the statistical "distance" between two distributions and is related to the mutual information between the two underlying variables (Cover and Thomas, 1991). These algorithms typically deliver very good estimates of the sources but can be very expensive in terms of computational resources and are not always robust against noise and outliers because of the need to estimate high order statistical cumulants based on a finite number of samples. Moreover, these methods require non-gaussian sources, because gaussian signals do not

have statistics with order higher than 2. Furthermore, dependence across time may not be captured: a white noise signal and the same signal shifted in time appear to be independent with respect to higher order measures.

Here we focus on algorithms of class (ii). In this case two signals are considered independent if each signal is decorrelated with time shifted copies of the other signal. For this class of BSS algorithms second-order statistics is sufficient (see e.g. Tong, Liu, Soon, and Huang, 1991; Molgedey and Schuster, 1994). We describe a method introduced by Molgedey and Schuster (1994) based only on second-order statistics.

An objective function must be defined that measures the mutual dependence of the estimated source components u_i as a function of the orthogonal transformation $\mathbf{Q}$. Finding the global minimum of the objective function with respect to $\mathbf{Q}$ yields the optimal transformation, which applied to the mixture returns the estimated sources. The correlation matrix of the estimated sources with time delay τ is defined by:

$$\mathbf{C}^{(\mathbf{u})}(\tau) := \langle \mathbf{u}(t)\mathbf{u}(t+\tau)^{\mathrm{T}} \rangle \tag{2.6}$$

If the components of $\mathbf{u}$ are mutually independent, the matrices $\mathbf{C}^{(\mathbf{u})}$ must be diagonal for all τ. We use a symmetrized form of the correlation matrix, which has the useful property of being diagonalizable with real eigenvalues and eigenvectors:

$$\mathbf{C}^{(\mathbf{u})}(\tau) := \frac{1}{2}\langle \mathbf{u}(t)\mathbf{u}(t+\tau)^{\mathrm{T}} + \mathbf{u}(t+\tau)\mathbf{u}(t)^{\mathrm{T}} \rangle , \tag{2.7}$$

$$C_{ij}^{(\mathbf{u})}(\tau) := \frac{1}{2}\langle u_i(t)u_j(t+\tau) + u_i(t+\tau)u_j(t) \rangle , \tag{2.8}$$

where $C_{ij}^{(\mathbf{u})}(\tau)$ is an entry of the symmetrized time delayed correlation matrix. The objective function can then be defined as:

$$\Psi^{\tau}{}_{\mathrm{ICA}}(\mathbf{Q}) := \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \left(C_{ij}^{(\mathbf{u})}(\tau) \right)^2 = \sum_{\substack{i,j=1 \\ i\neq j}}^{N} \left(\sum_{k,l=1}^{N} Q_{ik}Q_{jl}C_{kl}^{(\mathbf{y})}(\tau) \right)^2 \tag{2.9}$$

operating on the already whitened signal $\mathbf{y}(t)$. Minimization of $\Psi^{\tau}{}_{\mathrm{ICA}}$ can then be understood intuitively as finding an orthogonal matrix $\mathbf{Q}$ that diagonalizes the correlation matrix with time delay τ. Since, because of the whitening, the instantaneous correlation matrix, which is simply the covariance matrix, is already diagonal, this results in signal components that are decorrelated instantaneously and at a given time delay τ. This can be sufficient to achieve statistical independence (Tong et al., 1991). The time delay τ must, hovewer, be chosen such that all the eigenvalues of the correlation matrix are distinct. A more robust approach consists in extending this method to several time delays, thus reducing the problem to the joint-diagonalization of several correlation matrices (see e.g. Belouchrani, Abed Meraim, Cardoso, and Éric Moulines, 1997; Ziehe and

Müller, 1998). The corresponding objective function can be written as:

$$\Psi_{\text{ICA}}(\mathbf{Q}) := \sum_{\tau \in T} \Psi^{\tau}_{\text{ICA}}(\mathbf{Q}) \tag{2.10}$$

where T is the set of time delays. The optimization procedure consists in a sequence of plane rotations (Givens rotations) as described by Cardoso and Souloumiac (1996) and in Section 3.1.2.

2.2 Slow Feature Analysis

Slow Feature Analysis (SFA) is a method that extracts slowly varying signals from a given observed signal (Wiskott, 1998; Wiskott and Sejnowski, 2002). This section gives a short description of the method as well as a link between SFA and second-order ICA (Blaschke et al., 2006).

Consider a vectorial input signal $\mathbf{x}(t) = [x_1(t), \ldots, x_M(t)]^{\mathrm{T}}$. The objective of SFA is to find an input-output function $\mathbf{g}(\mathbf{x}) = [g_1(\mathbf{x}), \ldots, g_L(\mathbf{x})]^{\mathrm{T}}$ such that the components of $\mathbf{u}(t) = \mathbf{g}(\mathbf{x}(t))$ are varying as slowly as possible. The functions $g_i(\mathbf{x})$ can be in general non-linear. As a measure of slowness we use the variance of the first derivative, so that a slow signal has on average a small slope. The optimization problem consists in minimizing the objective function

$$\Delta(u_i) := \langle \dot{u}_i^2(t) \rangle \tag{2.11}$$

successively for each $u_i(t)$ under the constraints

$$\langle u_i(t) \rangle = 0 \qquad \text{(zero mean)}, \tag{2.12}$$

$$\langle (u_i(t))^2 \rangle = 1 \qquad \text{(unit variance)}, \tag{2.13}$$

$$\langle u_i(t) u_j(t) \rangle = 0 \quad \forall\, j < i \qquad \text{(decorrelation and order)}, \tag{2.14}$$

where $\langle \cdot \rangle$ denotes averaging over time. Constraints (2.12) and (2.13) ensure that the solution is not the trivial solution $u_i(t) =$ const. Constraint (2.14) provides uncorrelated output signal components and thus guarantees that different components carry different information.

To make the optimization problem easier to solve we consider the components g_i of the input-output function to be a linear combination of a finite set of nonlinear functions. We can then split the optimization procedure into two parts: (i) nonlinear expansion of the input signal $\mathbf{x}(t)$ into a high-dimensional feature space, and (ii) solving the optimization problem in this feature space linearly.

2.2.1 Nonlinear expansion

A common method to make nonlinear problems solvable in a linear fashion is nonlinear expansion. The observed signal components $x_i(t)$ are mapped into a high-dimensional feature-space according to

$$\mathbf{z}(t) = \mathbf{h}(\mathbf{x}(t)). \tag{2.15}$$

The dimension L of $\mathbf{z}(t)$ is typically much larger than that of the original signal. For instance, if we want to expand into the space of second degree polynomials, we can apply the mapping

$$\mathbf{h}(\mathbf{x}) = [x_1, \ldots, x_M, x_1x_1, x_1x_2, \ldots, x_Mx_M]^{\mathrm{T}} - \mathbf{h}_0{}^{\mathrm{T}}. \tag{2.16}$$

The dimensionality of this feature space is $L = M + M(M+1)/2$. The constant vector $\mathbf{h}_0{}^{\mathrm{T}}$ is needed to make the expanded signal mean free.

2.2.2 Solution of the linear optimization problem

Given the nonlinear expansion, the nonlinear input-output function $\mathbf{g}(\mathbf{x})$ can be written as

$$\mathbf{g}(\mathbf{x}) = \mathbf{R}\mathbf{h}(\mathbf{x}) = \mathbf{R}\mathbf{z}\,, \tag{2.17}$$

where $\mathbf{R}$ is an $L \times L$ matrix which is subject to optimization. To simplify the optimization procedure we (i) choose the nonlinearities $\mathbf{h}(\cdot)$ such that $\mathbf{z}(t)$ is mean free and (ii) first find a transformation $\mathbf{y}(t) = \mathbf{W}\mathbf{z}(t)$ to obtain mutually decorrelated components $y_i(t)$ with zero mean. Matrix $\mathbf{W}$ is a whitening matrix as in standard ICA:

$$\mathbf{u}(t) = \mathbf{Q}\mathbf{y}(t) = \mathbf{Q}\mathbf{W}\mathbf{z}(t) = \mathbf{R}\mathbf{z}(t) = \mathbf{g}(\mathbf{x}(t))\,, \tag{2.18}$$

where $\mathbf{y}(t)$ is the nonlinearly expanded and whitened signal. It can be shown (Wiskott and Sejnowski, 2002) that the constraints (2.12), (2.13), and (2.14) are fulfilled trivially if the transformation $\mathbf{Q}$, subject to learning, is an orthogonal matrix. To solve the optimization problem we rewrite the slowness objective (2.11)

$$\Delta(u_i) = \langle (\dot{u}_i(t))^2 \rangle = \mathbf{q}_i{}^{\mathrm{T}} \langle \dot{\mathbf{y}}(t)\, \dot{\mathbf{y}}(t)^{\mathrm{T}} \rangle \mathbf{q}_i =: \mathbf{q}_i{}^{\mathrm{T}} \mathbf{E} \mathbf{q}_i\,, \tag{2.19}$$

where $\mathbf{q}_i = [Q_{i1}, Q_{i2}, \ldots, Q_{iL}]^{\mathrm{T}}$ is the i-th row of $\mathbf{Q}$ and $\mathbf{E}$ is the matrix $\langle \dot{\mathbf{y}}(t) \dot{\mathbf{y}}(t)^{\mathrm{T}} \rangle$. For this optimization problem there exists a generally unique solution. For $i = 1$ the optimal weight vector is the normalized eigenvector that corresponds to the smallest eigenvalue of $\mathbf{E}$. The eigenvectors of the next higher eigenvalues produce the next slow components $u_2(t)$, $u_3(t)$, ... and so forth. Typically only the first several of all L possible output components are of interest and selected.

Finding the eigenvectors is equivalent to finding the transformation $\mathbf{Q}$ such that $\mathbf{Q}^{\mathrm{T}}\mathbf{E}\mathbf{Q}$ is diagonal. As described in detail in (Blaschke et al., 2006), this leads to

an objective function for SFA subject to maximization

$$\Psi^{\tau}{}_{\mathrm{SFA}}(\mathbf{Q}) := \sum_{i=1}^{L} \left(C_{ii}^{(\mathbf{u})}(\tau) \right)^2 = \sum_{i=1}^{L} \left(\sum_{k,l=1}^{L} Q_{ik} Q_{il} C_{kl}^{(\mathbf{y})}(\tau) \right)^2 , \tag{2.20}$$

where τ is a time delay that arises from an approximation of the time derivative. We set $\tau = 1$ because we make the approximation $\dot{\mathbf{y}}(t) \approx \mathbf{y}(t+1) - \mathbf{y}(t)$.

To understand (2.20) intuitively we note that slowly varying signal components are easier to predict and should therefore have strong correlations in time. Thus, maximizing the time delayed auto-correlation produces a slowly varying signal component. Since the trace of $\mathbf{C}^{(\mathbf{y})}(\tau)$ is preserved under a rotation $\mathbf{Q}$, maximizing the sum over the squared auto-correlations tends to produce a set of most slowly varying signal components at the expense of the other components, which become most quickly varying and are usually discarded.

Note the formal similarity between (2.9) and (2.20).

2.2.3 ICA and linear SFA

If we restrict ourselves to the case where we search for *linear* input-output functions $\mathbf{g}(\mathbf{x})$ for SFA, an interesting insight into the similarity between SFA and ICA can be obtained (Blaschke et al., 2006). In the linear case the nonlinear expansion step depicted in Section 2.2.1 can be skipped, and $L = N$. We have then that the input-output functions $\mathbf{g}(\mathbf{x})$ can be written as

$$\mathbf{g}(\mathbf{x}) = \mathbf{Q}\mathbf{y} = \mathbf{Q}\mathbf{W}\mathbf{x} \tag{2.21}$$

where $\mathbf{W}$ is a whitening matrix and $\mathbf{y}$ the whitened input signal. The equations derived in the previous paragraph retain their vailidity, and the linear SFA objective then reads:

$$\Psi_{\ \mathrm{SFA}}(\mathbf{Q}) := \sum_{i=1}^{N} \left(C_{ii}^{(\mathbf{u})}(1) \right)^2 = \sum_{i=1}^{N} \left(\sum_{k,l=1}^{N} Q_{ik} Q_{il} C_{kl}^{(\mathbf{y})}(1) \right)^2 , \tag{2.22}$$

where we have already set $\tau = 1$. Note that because the sum of the squared entries of a matrix is conserved under an orthogonal transformation

$$\sum_{i,j} \left(C_{ij}^{(\mathbf{u})}(1) \right)^2 = \sum_{i,j} \left(C_{ij}^{(\mathbf{y})}(1) \right)^2 = \mathrm{const}, \tag{2.23}$$

we can split the previous sum into two terms

$$\sum_{i,j} \left(C_{ij}^{(\mathbf{u})}(1) \right)^2 = \sum_{i} \left(C_{i}^{(\mathbf{u})}(1) \right)^2 + \sum_{i \neq j} \left(C_{ij}^{(\mathbf{u})}(1) \right)^2 = \mathrm{const}, \tag{2.24}$$

and intuitively understand that maximization of Ψ_{SFA} in Eq. (2.22) is equivalent to minimization of the objective Ψ^{τ}_{ICA} in Eq. (2.9) when $\tau = 1$. This observation leads us to the important result that **linear SFA is formally equivalent to second-order ICA with time delay one**.

2.3 Nonlinear BSS and ICA

An obvious extension to the linear mixing model (2.4) has the form

$$\mathbf{x}(t) = \mathcal{F}(\mathbf{s}(t)) \;, \tag{2.25}$$

with a nonlinear function $\mathcal{F} : \mathbb{R}^N \to \mathbb{R}^M$ that maps N-dimensional source vectors $\mathbf{s}(t)$ onto M-dimensional signal vectors $\mathbf{x}(t)$. The components $x_i(t)$ of the observable are a nonlinear mixture of the sources and like in the linear case source signal components $s_i(t)$ are assumed to be mutually statistically independent. Extracting the source signal is of course only possible if $\mathcal{F}$ is an invertible function on the range of $\mathbf{s}(t)$, which we assume from now on.

The equivalence of BSS and ICA in the linear case does not hold in general for a nonlinear function $\mathcal{F}$ (Hyvärinen and Pajunen, 1999; Jutten and Karhunen, 2003). For example, given statistically independent components $s_1(t)$ and $s_2(t)$, any nonlinear functions $h_1(s_1)$ and $h_2(s_2)$ also lead to components that are statistically independent, so there is an infinite number of independent signal pairs of the form $\mathbf{h}(\mathbf{s})$ that could give rise to the same observed mixture $\mathbf{x}(t)$. Also, a nonlinear mixture of $s_1(t)$ and $s_2(t)$ can still have statistically independent components (Jutten and Karhunen, 2003). An example of the latter can be easily constructed. Assume we have two independent sources s_1 and s_2, where $s_1 \in \mathbb{R}^+$ with a probability density function $p_{s_1}(s_1) = s_1 \mathrm{e}^{-s_1^2/2}$ and s_2 is uniformly distributed in $[0, 2\pi)$. If we apply the nonlinear mapping

$$\begin{aligned}[y_1, y_2] &= H(s_1, s_2) \\ &= [s_1 \cos(s_2), s_1 \sin(s_2)],\end{aligned} \tag{2.26}$$

with a Jacobian matrix

$$\mathbf{J} = \begin{pmatrix} \cos(s_2) & -s_1 \sin(s_2) \\ \sin(s_2) & s_1 \cos(s_2)\,, \end{pmatrix}. \tag{2.27}$$

the joint probability density function of y_1 and y_2 is

$$\begin{aligned} p_{y_1,y_2}(y_1, y_2) &= \frac{p_{s_1,s_2}(s_1, s_2)}{|\mathbf{J}|} \\ &= \frac{1}{2\pi} \mathrm{e}^{(-y_1^2 - y_2^2)/2} \\ &= \left(\frac{1}{\sqrt{2\pi}} \mathrm{e}^{-y_1^2/2}\right) \left(\frac{1}{\sqrt{2\pi}} \mathrm{e}^{-y_2^2/2}\right) \\ &= p_{y_1}(y_1) \cdot p_{y_2}(y_2)\,. \end{aligned} \tag{2.28}$$

As the joint distribution is separable, the two random variables y_1 and y_2 are statistically independent, even if they are a nonlinear mixture of the sources.

This means that in the nonlinear BSS problem independence is not sufficient to recover the original source signal and additional assumptions about the mapping $\mathcal{F}$ or the source signal are needed to sufficiently constrain the optimization problem. Typical assumptions include:

- constraints on the mapping $\mathcal{F}$:
  - $\mathcal{F}$ is a smooth mapping (Hyvärinen and Pajunen, 1999; Almeida, 2004);
  - $\mathcal{F}$ is a post nonlinear (PNL) mapping (Taleb and Jutten, 1997; Yang, Amari, and Cichocki, 1998; Taleb and Jutten, 1999; Taleb, 2002; Ziehe, Kawanabe, Harmeling, and Müller, 2003).
- prior information about the source signal components:
  - source signal components are bounded (Babaie-Zadeh, Jutten, and Nayebi, 2002);
  - source signal components have time-delayed auto-correlations (referred to as temporal correlations) (Hosseini and Jutten, 2003);
  - source signal components are those that exhibit a characteristic time structure (power spectra are pairwise different) (Harmeling, Ziehe, Kawanabe, and Müller, 2003).

In the next chapter we expose an approach that does not require specific assumptions on the nonlinear mapping $\mathcal{F}$.

3 Independent Slow Feature Analysis

In this chapter we present a novel approach to the nonlinear blind source separation problem in which no specific assumption about the mapping F in Eq. (2.25) needs to be made, although the function space available for unmixing is finite-dimensional in the algorithm, which imposes some limitations on F. Since we employ an ICA method based on time-delayed cross-correlations we make the implicit assumption that the sources have significantly different temporal structure (power spectra are pairwise different) (cf. Harmeling et al., 2003). We also assume that the sampling rate is high enough, so that the input signal can be treated as if it were continuous and the time derivative is well approximated by the difference of two successive time points.

We have seen above that in the nonlinear case statistical independence alone is not a sufficient criterion for blind source separation. There are infinitely many nonlinearly distorted versions of one source that are all statistically independent of another source. We propose slowness as a means to resolve this ambiguity and select a good representative from all the different versions of a source, because nonlinearly distorted versions of a source are usually varying more quickly than the source itself. Consider for example a sinusoidal signal component $x_i(t) = \sin(t)$ and a second component that is the square of the first $x_j(t) = x_i(t)^2 = 0.5\,(1 - \cos(2t))$. The second component is more quickly varying due to the frequency doubling induced by the squaring. This argument can be made more formal and it can be proven that, given the set of a one-dimensional signal and all its nonlinearly and continuously transformed versions, the slowest signal of the set is either the signal itself or an invertibly transformed version of it: a formal proof is sketched in Chapter 4. Considering this we propose, in order to perform nonlinear BSS, to complement the independence objective of pure ICA with a slowness objective. The work presented is this chapter is joint work with Dr. Tobias Blaschke and Prof. Laurenz Wiskott and has been published in Blaschke, Zito, and Wiskott (2007).

3.1 Independent Slow Feature Analysis

The nonlinear BSS method proposed in this chapter combines the principle of independence known from linear second-order BSS methods with the principle of slowness as described above. Because of the combination of ICA and SFA we refer to this method as Independent Slow Feature Analysis (ISFA). As already explained, second-order ICA tends to make the output components independent and SFA tends to make them slow. Since we are dealing with a nonlinear mixture we first compute a nonlinearly expanded

signal $\mathbf{z}(t) = \mathbf{h}(\mathbf{x}(t))$ with $\mathbf{h} : \mathbb{R}^M \to \mathbb{R}^L$ being some nonlinear function chosen such that $\mathbf{z}(t)$ has zero mean. In a second step $\mathbf{z}(t)$ is whitened to obtain $\mathbf{y}(t) = \mathbf{W}\mathbf{z}(t)$. Finally we apply linear ICA combined with linear SFA on $\mathbf{y}(t)$ in order to find the output signal $\mathbf{u}(t)$, the R first component of which are the estimated source signals, where R is usually much smaller than L, the dimension of the expanded signal. Because of the whitening we know that ISFA, like ICA and SFA, is solved by finding an orthogonal $L \times L$ matrix $\mathbf{Q}$. We write the output signal $\mathbf{u}(t)$ as

$$\mathbf{u}(t) = \mathbf{Q}\mathbf{y}(t) = \mathbf{Q}\mathbf{W}\mathbf{z}(t) = \mathbf{Q}\mathbf{W}\mathbf{h}(\mathbf{x}(t)) \,. \tag{3.1}$$

While the $u_1(t), \ldots, u_R(t)$ are statistically independent and slowly varying, the components $u_{R+1}(t), \ldots, u_L(t)$ are more quickly varying and may be statistically dependent on each other as well as on the estimated sources. The last $L - R$ components of the output signal $\mathbf{u}(t)$ are irrelevant for the final result but important during the optimization procedure, see below.

To summarize, we have an M-dimensional input $\mathbf{x}(t)$, an L-dimensional nonlinearly expanded and whitened $\mathbf{y}(t)$, and an L-dimensional output signal $\mathbf{u}(t)$. ISFA finds an orthogonal matrix $\mathbf{Q}$ such that the R first components of the output signal $\mathbf{u}(t)$ are mutually independent and slowly varying. These are the estimated sources.

3.1.1 Objective function

To recover R source signal components $u_i(t), \;\; i = 1, \ldots, R$ from an L-dimensional expanded and whitened signal $\mathbf{y}(t)$ the objective for ISFA with one single time delay τ reads

$$\begin{aligned} \Psi^{\tau}{}_{\mathrm{ISFA}}(u_1, \ldots, u_R) &:= b_{\,\mathrm{ICA}} \Psi^{\tau}{}_{\mathrm{ICA}}(u_1, \ldots, u_R) - b_{\,\mathrm{SFA}} \Psi^{\tau}{}_{\mathrm{SFA}}(u_1, \ldots, u_R) \\ &= b_{\,\mathrm{ICA}} \sum_{\substack{i,j=1,\\ i \neq j}}^{R} \left(C_{ij}^{(\mathbf{u})}(\tau) \right)^2 - b_{\,\mathrm{SFA}} \sum_{i=1}^{R} \left(C_{ii}^{(\mathbf{u})}(\tau) \right)^2 , \end{aligned} \tag{3.2}$$

where we simply combine the ICA objective (2.9) and SFA objective (2.20) for the first R components weighted by the factors $b_{\,\mathrm{ICA}}$ and $b_{\,\mathrm{SFA}}$, respectively. Note that the ICA and the SFA objective are usually applied to all components and that in the linear case (and for one time delay $\tau = 1$) they are equivalent (see Section 2.2.3). Here, they are applied to an R-dimensional subspace in the L-dimensional expanded space, which makes them different from each other. $\Psi^{\tau}{}_{\mathrm{ISFA}}$ is to be minimized, which is the reason why the SFA part has a negative sign.

In the linear case it is standard practice to use multiple time delays to stabilize the ICA solution, see for example the kTDSEP algorithm by Harmeling et al. (2003). We see in Sections 3.2.1 and 3.2.2 that in our case multiple time-delays are actually essential to get meaningful solutions. The general expression for the objective of ISFA then reads

$$\begin{aligned}\Psi_{\text{ ISFA}}(u_1,\ldots,u_R) &:= b_{\text{ ICA}} \sum_{\tau\in T_{\text{ ICA}}} \kappa^{\tau}_{\text{ICA}} \Psi^{\tau}_{\text{ICA}} - b_{\text{ SFA}} \sum_{\tau\in T_{\text{ SFA}}} \kappa^{\tau}_{\text{SFA}} \Psi^{\tau}_{\text{SFA}} \\ &= b_{\text{ ICA}} \sum_{\tau\in T_{\text{ ICA}}} \kappa^{\tau}_{\text{ICA}} \sum_{\substack{i,j=1,\\ i\neq j}}^{R} \left(C^{(\mathbf{u})}_{ij}(\tau)\right)^2 \\ &\quad - b_{\text{ SFA}} \sum_{\tau\in T_{\text{ SFA}}} \kappa^{\tau}_{\text{SFA}} \sum_{i=1}^{R} \left(C^{(\mathbf{u})}_{ii}(\tau)\right)^2 ,\end{aligned} \tag{3.3}$$

where $T_{\text{ ICA}}$ and $T_{\text{ SFA}}$ are the sets of time delays for the ICA and SFA objectives, respectively, whereas $\kappa^{\tau}_{\text{ICA}}$ and $\kappa^{\tau}_{\text{SFA}}$ are weighting factors for the corresponding correlation matrices.

For simplicity we first continue the description with only one time delay based on (3.2) and only later provide the full formulation with multiple time delays based on (3.3).

3.1.2 Optimization procedure

From (3.1) we know that $\mathbf{C}^{(\mathbf{u})}(\tau)$ in (3.2) depends on the orthogonal matrix $\mathbf{Q}$. There are several ways to find the orthogonal matrix that minimizes the objective function. Here we apply successive Givens rotations to obtain $\mathbf{Q}$. A Givens rotation is a rotation around the origin within the plane of two selected components μ and ν and has the matrix form

$$Q^{\mu\nu}_{ij} := \begin{cases} \cos(\phi) & \text{for } (i,j) \in \{(\mu,\mu),(\nu,\nu)\} \\ -\sin(\phi) & \text{for } (i,j) \in \{(\mu,\nu)\} \\ \sin(\phi) & \text{for } (i,j) \in \{(\nu,\mu)\} \\ \delta_{ij} & \text{otherwise} \end{cases} \tag{3.4}$$

with Kronecker symbol δ_{ij} and rotation angle ϕ. Any orthogonal $L\times L$ matrix such as $\mathbf{Q}$ can be written as a product of $L(L-1)/2$ (or more) Givens rotation matrices $\mathbf{Q}^{\mu\nu}$ (for the rotation part) and a diagonal matrix with diagonal elements ± 1 (for the reflection part). Since reflections do not matter in our case we only consider the Givens rotations as is often done in second-order ICA algorithms (e.g. Cardoso and Souloumiac, 1996). Givens rotations are applied to the whole space, but the objective function only takes into account a subspace. The objective (3.2) as a function of a Givens rotation $\mathbf{Q}^{\mu\nu}$

reads

$$\begin{aligned}\Psi^{\tau,\mu\nu}_{\text{ISFA}}\left(\mathbf{Q}^{\mu\nu}\right) = b_{\text{ ICA}} \sum_{\substack{i,j=1\\ i\neq j}}^{R} \left(\sum_{k,l=1}^{L} Q^{\mu\nu}_{ik} Q^{\mu\nu}_{jl} C^{(\mathbf{u}')}_{kl}\left(\tau\right)\right)^2 \\ - b_{\text{ SFA}} \sum_{i=1}^{R} \left(\sum_{k,l=1}^{L} Q^{\mu\nu}_{ik} Q^{\mu\nu}_{il} C^{(\mathbf{u}')}_{kl}\left(\tau\right)\right)^2 ,\end{aligned} \tag{3.5}$$

where $\mathbf{u}'$ is some intermediate signal during the optimization procedure. For each Givens rotation there exists an angle $\phi_{\text{ min}}$ with minimal $\Psi^{\tau,\mu\nu}_{\text{ISFA}}$. Successive application of many Givens rotations $\mathbf{Q}^{\mu\nu}$ with their corresponding rotation angle $\phi_{\text{ min}}$ leads to the final rotation matrix $\mathbf{Q}$ yielding

$$\mathbf{C}^{(\mathbf{u})}\left(\tau\right) = \mathbf{Q}^{\text{ T}}\mathbf{C}^{(\mathbf{y})}\left(\tau\right)\mathbf{Q}\,. \tag{3.6}$$

In the ideal case the upper left $R \times R$ submatrix of $\mathbf{C}^{(\mathbf{u})}\left(\tau\right)$ is diagonal with a large trace $\sum_{i=1}^{R} C^{(\mathbf{u})}_{ii}\left(\tau\right)$.

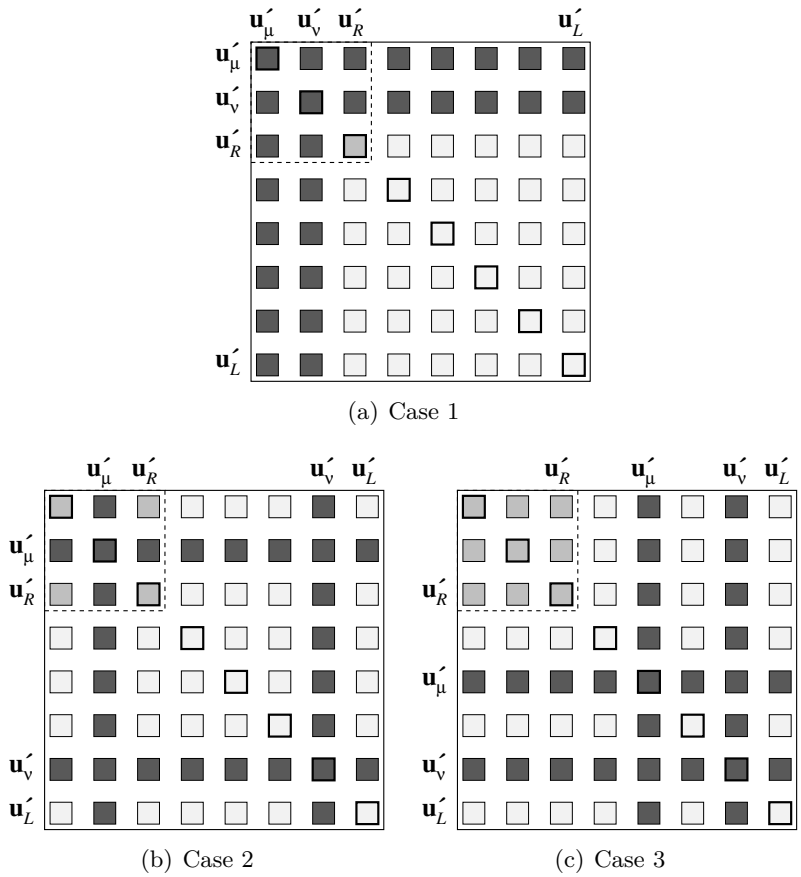
Applying a Givens rotation $\mathbf{Q}^{\mu\nu}$ in the $\mu\nu$-plane changes all auto- and cross-correlations $C^{(\mathbf{u}')}_{ij}\left(\tau\right)$ with at least one of the indices equal to μ or ν. There exist two invariances under such a transformation, which can be described as

$$\left(C^{(\mathbf{u}')}_{\mu i}\left(\tau\right)\right)^2 + \left(C^{(\mathbf{u}')}_{\nu i}\left(\tau\right)\right)^2 = \text{const } \forall i \notin \{\mu,\nu\}, \tag{3.7}$$

$$\left(C^{(\mathbf{u}')}_{\mu\mu}\left(\tau\right)\right)^2 + \left(C^{(\mathbf{u}')}_{\mu\nu}\left(\tau\right)\right)^2 + \left(C^{(\mathbf{u}')}_{\nu\mu}\left(\tau\right)\right)^2 + \left(C^{(\mathbf{u}')}_{\nu\nu}\left(\tau\right)\right)^2 = \text{const}. \tag{3.8}$$

Invariance (3.7) can be understood as the invariance of a vector's norm under rotation, whereas (3.8) states again the invariance of the sum of the squared entries of a matrix under an orthogonal transformation already introduced in Equation (2.23). Assume we want to minimize $\Psi^{\tau}_{\text{ISFA}}$ for a given R, where R denotes the number of signal components we want to extract. Applying a Givens rotation $\mathbf{Q}^{\mu\nu}$ we have to distinguish three cases:

Case 1 Both axes, μ and ν, lie inside the subspace spanned by the first R axes ($\mu, \nu \leq R$) (see Fig. 3.1a): The sum over all squared cross correlations of all signal components that lie outside the R-dimensional subspace is constant as well as that of all signal components inside the subspace. The former holds because of the first invariance (3.7) and the latter because of the first (3.7) and second invariance (3.8). There is no interaction between inside and outside, in fact the objective function is exactly the objective for an ICA algorithm based on second-order statistics, e.g. TDSEP or SOBI (Ziehe and Müller, 1998; Belouchrani et al., 1997). In (Blaschke et al., 2006) it has been shown that this is equivalent to SFA in the case of a single time



(a) Case 1

(b) Case 2

(c) Case 3

Figure 3.1: Objective function Ψ^τ_{ISFA} minimization. Each square represents a squared cross or auto-correlation $\left(C_{ij}^{(\mathbf{u}')}\right)^2$ where index $i\,(j)$ denotes the row (column) of the square. Dark squares indicate all entries that are changed by a rotation in the μ-ν-plane. L is the dimensionality of the expanded signal $\mathbf{u}'$ and R the number of signal components $u'_i(t)$ subject to optimization. The entries incorporated in the objective function are located in the upper left corner as indicated by the dashed line. Figure from Blaschke et al. (2007).

delay of $\tau = 1$.

Case 2 Only one axis, w.l.o.g. μ, lies inside the subspace; the other, ν, lies outside ($\mu \leq R < \nu$) (see Fig. 3.1b): Since one axis of the rotation plane lies outside the subspace, u'_μ in the objective function can be optimized at the expense of the u'_ν outside the subspace. A rotation of $\pi/2$, for example, would simply exchange components u'_μ and u'_ν. For instance, according to (3.7) $\left(C_{\mu i}^{(\mathbf{u}')}\right)^2$ can be optimized

at the expense of $\left(C^{(\mathbf{u}')}_{\nu i}\right)^2$ with $i \in \{1, \ldots, R\}$; according to (3.8) $\left(C^{(\mathbf{u}')}_{\mu\mu}\right)^2$ can be optimized at the expense of $\left(C^{(\mathbf{u}')}_{\mu\nu}\right)^2$, $\left(C^{(\mathbf{u}')}_{\nu\mu}\right)^2$, and $\left(C^{(\mathbf{u}')}_{\nu\nu}\right)^2$. This gives the possibility to find the slowest and most independent components in the whole space spanned by all L axes in contrast to Case 1 where the minimum is searched within the subspace spanned by the first R axes considered in the objective function.

Case 3 Both axes lie outside the subspace ($R < \mu, \nu$) (see Fig. 3.1c): A Givens rotation with the two rotation axes outside the relevant subspace does not affect the objective function and can therefore be disregarded.

To optimize the objective function of ISFA (3.2) we need to calculate the explicit form of the objective function $\Psi^{\tau,\mu\nu}_{\text{ISFA}}$ in (3.5). By inserting the Givens rotation matrix (3.4) into the objective function (3.5), and considering the case with multiple time delays, we can write the objective as a function of the rotation angle ϕ

$$\begin{aligned}\Psi^{\mu\nu}_{\text{ISFA}}(\phi) = b_{\text{ICA}} &\left(e_c + \sum_{\beta=0}^{2} e_\beta \cos^{4-\beta}(\phi)\sin^{\beta}(\phi)\right) \\ - b_{\text{SFA}} &\left(d_c + \sum_{\alpha=0}^{4} d_\alpha \cos^{4-\alpha}(\phi)\sin^{\alpha}(\phi)\right)\end{aligned} \tag{3.9}$$

with constants e and d that depend only on the $C^{(\mathbf{u}')}_{kl}$ before rotation. Further simplification (cf. Blaschke and Wiskott, 2004) leads to

$$\text{Case 1: } \Psi^{\mu\nu}_{\text{ISFA}}(\phi) = A_0 + A_4\cos(4\phi + \phi_4) \tag{3.10}$$

$$\text{Case 2: } \Psi^{\mu\nu}_{\text{ISFA}}(\phi) = A_0 + A_2\cos(2\phi + \phi_2) + A_4\cos(4\phi + \phi_4) \tag{3.11}$$

with a single minimum (if w.l.o.g. $\phi \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$), which can be calculated easily. The derivation of (3.9), (3.10) and (3.11) involves various trigonometric identities and, because of its length, is documented in the appendix on page 79.

The iterative optimization procedure with successive Givens rotations can now be described as follows:

1. Initialize $\mathbf{Q}' = \mathbf{I}$ and compute $\mathbf{C}^{(\mathbf{u}')}(\tau) = \mathbf{C}^{(\mathbf{y})}(\tau)\,\forall\tau \in T_{\text{ICA}} \cup T_{\text{SFA}}$ with (2.7) and Ψ'_{ISFA} with (3.3).

2. Choose a random permutation of the set of axis pairs:
   $P = \sigma\left(\{(\mu,\nu), \text{ with } \mu \le R \text{ and } \mu < \nu \le L\}\right)$.

3. Go systematically through all axis pairs in P. For each axis pair:
   a) determine the optimal rotation angle $\phi^{\mu\nu}_{\min}$ for the selected axes with (3.10) or (3.11),

b) compute the Givens rotation matrix $\mathbf{Q}^{\mu\nu}\left(\phi^{\mu\nu}_{\min}\right)$ defined by (3.4),

c) update $\mathbf{C}^{(\mathbf{u}')}(\tau)$ using $\mathbf{C}^{(\mathbf{u}')}(\tau) \rightarrow \left(\mathbf{Q}^{\mu\nu}\right)^{\mathrm{T}} \mathbf{C}^{(\mathbf{u}')}(\tau)\, \mathbf{Q}^{\mu\nu}$,

d) update $\mathbf{Q}'$ according to $\mathbf{Q}' \rightarrow \mathbf{Q}^{\mu\nu}\mathbf{Q}'$,

e) backup the previous objective-function value $\Psi''_{\mathrm{ISFA}} = \Psi'_{\mathrm{ISFA}}$,

f) calculate the new objective-function value Ψ'_{ISFA} with (3.3) using the updated $\mathbf{C}^{(\mathbf{u}')}(\tau)$ from (3c),

g) store the relative decrease of the objective function value:

$$\frac{\Psi''_{\mathrm{ISFA}} - \Psi'_{\mathrm{ISFA}}}{|\Psi''_{\mathrm{ISFA}}|} .$$

4. Go to 2 until the relative decrease of the objective function is smaller than $\epsilon \ll 1$ for all axis pairs in P.

5. Set $\mathbf{Q} = \mathbf{Q}'$ and $\mathbf{u}(t) = \mathbf{Q}\mathbf{y}(t)$.

In Step 2 it is important to note that the rotation planes of the Givens rotations are selected from the whole L-dimensional space (although we avoid the irrelevant Case 3 by requiring $\mu \leq R$, see Fig. 3.1) whereas the objective function only uses information of correlations among the first R signal components u'_i. Since $\mathbf{Q}_{\mu\nu}$ is very sparse, the Givens rotation in Step 3c does not require a full matrix multiplication but can be computed more efficiently.

Note that the algorithm works on the intermediate correlation matrices $\mathbf{C}^{(\mathbf{u}')}(\tau)$ and not on the signals themselves; the input signal $\mathbf{y}(t)$ is used only in the initialization (Step 1) and at the end (Step 5) when the output signal $\mathbf{u}(t)$ is computed.

To circumvent the problem of getting stuck in local optima of the objective function, a random rotation of the outer space ($\nu > \mu > R$) can be performed after convergence in Step 4, and the algorithm can be restarted at Step 2.

3.2 Results

To evaluate the performance of ISFA we tested the algorithm first on random symmetric matrices to check how many matrices are needed to get meaningful results, then on surrogate matrices to check that the algorithm reliably converges to the global optimum under these ideal conditions, and then on a difficult although low-dimensional mixture of audio data to show how it performs on real data. In order to reduce the problem of local optima, we use SFA as a preprocessing step. That choice follows from the empirical observation that SFA is always able to extract the first source signal. To stabilize the ISFA solutions even further we typically run the optimization routine once with the

first axis fixed, and then once more following the procedure described in Section 3.1.2. Throughout the paper the SFA time-delay set and the weighting factors were as follows:

$$T_{\mathrm{SFA}} = \{1\} \tag{3.12}$$
$$\kappa^{\tau}_{\mathrm{SFA}} = 1 \quad \text{for } \tau = 1 \tag{3.13}$$
$$\kappa^{\tau}_{\mathrm{ICA}} = 1 \quad \forall\, \tau \in T_{\mathrm{ICA}}\,; \tag{3.14}$$

This particular choice makes it easy to interpret the ISFA objective function (3.3): The SFA part is the plain SFA objective function of (2.20); the ICA part is the plain ICA objective function of (2.9) extended to several time delays. If we would choose more than one time delay for the SFA part, the interpretation in terms of slowness would become less clear (see Blaschke et al., 2006). T_{ICA} depends on the experiment, see below.

3.2.1 Tests with random matrices

First consider only the ICA part of the objective function (3.3). Its purpose is to guarantee statistical independence of the estimated sources by simultaneously diagonalizing the $R \times R$ upper left submatrix of T time-delayed $L \times L$ correlation matrices, where T is the number of elements in T_{ICA}. However, for the ICA term to be useful we have to take sufficiently many matrices into account so that simultaneous submatrix-diagonalization is not trivial. For instance, a single symmetric matrix can always be fully diagonalized by the orthonormal set of its eigenvectors. Thus for $R = L$ and $T = 1$ one has to take at least two matrices to avoid this spurious solution, which would be found even if there are no underlying statistically independent sources.

To estimate the minimum number of matrices needed, we ran ISFA with $b_{\mathrm{SFA}} = 0$ on randomly generated symmetric matrices $\mathbf{A}^{\tau}$, $\tau = 1, ..., T$, for different values of L, R, and T. The subdiagonalization was considered successful if $E := \sqrt{\langle A_{ij}^2 \rangle_{\tau,j,i>j}}$, i.e. the square root of the averaged squared non-diagonal terms, was below a threshold $E_{\mathrm{crit}} := 10^{-3}$. For fixed L and $R < L$ we typically observe that a high degree of subdiagonalization is possible for $T = 2$. For $T > 2$ the subdiagonalization is still possible but at a lower degree with increasing T, until a critical T_{crit} is reached, for which the degree of subdiagonalization displays a sharp transition where E crosses the threshold E_{crit} and remains stable after that.

The estimated critical number of time delays T_{crit} for $L \in \{9, 20\}$ and different values of R are given in Table 3.1. In the simulations that follow, we have $M = R = 2$ and use ISFA^3 and ISFA^5 (ISFA^n refers to ISFA with polynomials of degree n) resulting in $L = 9$ and $L = 20$, respectively. From the table we see that with $T = 50$ we are well above T_{crit} in both cases.

	R	2	3	4	5	6	7	8	9	10	>10
$L = 9$	T_{crit}	18	8	5	4	3	2	2	2	-	-
$L = 20$	T_{crit}	36	19	13	9	7	6	5	4	3	2

Table 3.1: Critical number of time delays T_{crit} for different values of L and R.

3.2.2 Tests with surrogate matrices

To test the performance of ISFA (now including the SFA part) in absence of noise, finite-size effects, or any other kind of perturbation we carried out an experiment with $T > 1$ surrogate matrices, prepared such that they have a unique exact solution (except for permutations). The first matrix, with $\tau = 1$, is fully diagonal with the diagonal elements ordered by decreasing absolute value, with the exception of the second and last element, which are swapped. All other $T - 1$ matrices are random symmetric matrices with a diagonal $(R + R_{\text{ICA}}) \times (R + R_{\text{ICA}})$ upper submatrix. SFA alone only sees the first matrix (cf. 3.12, 3.13) and would favor a solution in which the last component is swapped back into the $R \times R$ subspace in place of the small second component. ICA alone would favor any permutation of the first $(R + R_{\text{ICA}})$ components equally well, because for any of these permutations the $R \times R$ upper submatrices are all diagonal. In this example ICA should prevent SFA from swapping the last component into the $R \times R$ subspace and SFA should disambiguate the many equally valid ICA solutions by selecting the largest diagonal elements, i.e. the slowest components, in the first matrix.

This set of matrices constitutes a fixed point for the ISFA algorithm. If we run ISFA directly on these matrices we get $\mathbf{Q} = \mathbf{I}$. If we now apply a random rotation matrix $\mathbf{Q}_{\text{rand}}$ to the set of matrices, we would expect ISFA to find a matrix $\mathbf{Q}$ that inverts this rotation and returns the R original first components, but in any arbitrary order. Thus, the $R \times R$ submatrix of the product $\mathbf{P} := \mathbf{Q}\mathbf{Q}_{\text{rand}}$ should be a permutation matrix within the $R \times R$ subspace for perfect unmixing.

We performed 10,000 independent tests with $R = 2$, $R_{\text{ICA}} = 2$, $L = 9$, and $T = 50$, somewhat imitating the case of two nonlinearly mixed independent sources and an expansion space of all polynomials of degree three. The estimated critical number of matrices T_{crit} is 18. Using 50 matrices we rule out any spurious solution as discussed in Section 3.2.1. As a measure of performance we used the reconstruction error measure first introduced by Amari, Cichocki, and Yang (1995) in the formulation given in (Blaschke and Wiskott, 2004):

$$E = \frac{1}{R^2}\left(\sum_{i=1}^{R}\left(\sum_{j=1}^{R}\frac{|P_{ij}|}{\max_k |P_{ik}|} - 1\right) + \sum_{j=1}^{R}\left(\sum_{i=1}^{R}\frac{|P_{ij}|}{\max_k |P_{kj}|} - 1\right)\right). \tag{3.15}$$

An experiment is considered to be successful if the unmixing error is smaller than 10^{-5}. We found that ISFA always recovered the original components and that this 100% success

rate was largely independent of the scaling factors b_{ICA} and b_{SFA}, which we therefore set to $b_{\text{ICA}} = b_{\text{SFA}} = 1$ for this experiment.

3.2.3 Tests with twisted audio data

In the third experiment we tested the algorithm on 171 pairs of 19 nonlinearly mixed music excerpts. The sample values of the 19 excerpts were in the range of $[-1, +1)$; the mean had an average value of $(-10 \pm 110) \times 10^{-6}$ (mean $\pm$ std); the standard deviation had an average value of 0.16 ± 0.07, its minimum and maximum value was 0.02 and 0.27, respectively. One additional music excerpt was discarded, because it had extreme peaks, which led to a strong nonlinear distortion due to the SFA part and low correlations with the source even though it was in principle extracted correctly. All audio signals were $2^{21} = 2,097,152$ samples long and had a CD-quality sampling frequency of 44,100 Hz. We used the nonlinear mixture introduced by Harmeling et al. (2003) defined by

$$x_1(t) = (s_2(t) + 3s_1(t) + 6)\cos(1.5\,\pi\, s_1(t))\,, \tag{3.16}$$

$$x_2(t) = (s_2(t) + 3s_1(t) + 6)\sin(1.5\,\pi\, s_1(t))\,. \tag{3.17}$$

This is quite an extreme nonlinearity and the unmixing performance depends strongly on the standard deviation of the sources. For the ICA part of the objective in (3.3) we used 50 time delays evenly spaced within 1 and 44,100, corresponding to a time scale up to 1 second. The number of time delays is greater than the critical number T_{crit}, which is 18 for an expansion with polynomials of degree three, and 36 for polynomials of degree five. In order to evaluate the performance of the algorithm fairly we used linear regression to check if the nonlinear mixture was indeed invertible within the available space. Two orthogonal directions were fit within the whitened expanded space to maximize the correlation with the original sources. Within the space of polynomials of degree three, there were a number of cases (51 examples, 30% of the total) where the two sources were not found by linear regression, which means the nonlinear mixture was not invertible within the available expanded space. This is the main reason for failures in ISFA[3]. Within the space of polynomials of degree five the mixture was always invertible. The scaling factor b_{SFA} was kept constant and equal to 1, while b_{ICA} was manually tuned for each example in order to maximize the correlation between estimated and original sources. For polynomials of degree three we tested different values of b_{ICA} equidistant on a logarithmic scale between 0 and 10000. The number of tested values varied between 5 and 40 depending on how clear and robust the optimum was. For polynomials of degree one and five we largely adopted the values found for polynomials of degree three; only if the algorithm failed with these values did we retune b_{ICA} with 20 equidistant values. This tuning resulted in values between 0 and 1000. A source signal is considered to be recovered if the correlation with the estimated source is greater than 0.9.

Scatter plots of a successful example are shown in Figure 3.2 and a summary of the

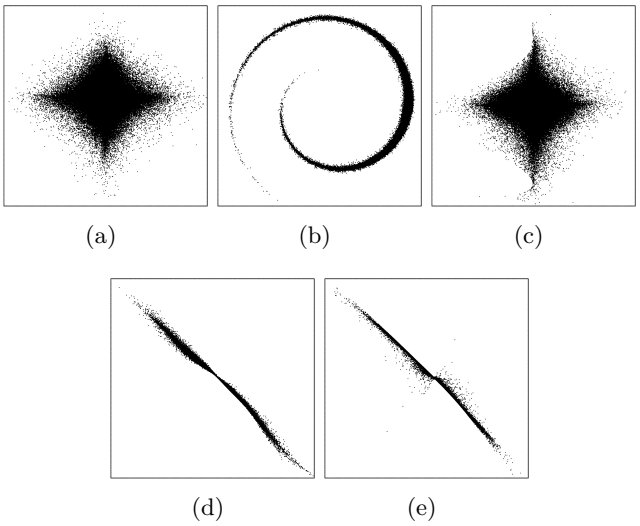


Figure 3.2: Scatter plot of two sources, their nonlinear mixture, and the estimated sources. (a) Sources, (b) mixture, (c) sources estimated by ISFA^5, (d) first source vs. estimated first source, (e) second source vs. estimated second source. Correlation coefficients of estimated sources and original sources were 0.996 and 0.998.

# rec. src.	REG^1		REG^3		REG^5	
2	5%	(8)	70%	(120)	100%	(171)
1	54%	(93)	30%	(51)	0%	(0)
0	41%	(70)	0%	(0)	0%	(0)
# rec. src.	ISFA^1		ISFA^3		ISFA^5	
2	5%	(8)	50%	(85)	71%	(122)
1	50%	(85)	34%	(59)	18%	(30)
0	45%	(78)	16%	(27)	11%	(19)
% correct	**100%**	$\left(\frac{8}{8}\right)$	**71%**	$\left(\frac{85}{120}\right)$	**71%**	$\left(\frac{122}{171}\right)$

Table 3.2: Summary of ISFA results. The upper part shows percentages of cases where both, one, or none of the two sources were recovered by linear regression (supervised) in the original space (REG^1) or in the expanded space with polynomials of degree three (REG^3) or five (REG^5). The lower part shows the same for ISFA (unsupervised except for the tuning of b_{ICA}). Each entry indicates the percentage (and number) of pairs with respect to the total of 171 pairs. The last line presents the percentage of both sources recovered correctly with respect to the number of mixtures invertible within the available expanded space by linear regression. Note that in the case of two recovered sources chance level is always smaller than 0.01%.

results is given in Table 3.2. ISFA is able to separate the nonlinearly mixed sources in about 70% of the cases in which unmixing was possible at all. This is remarkable given the extreme nonlinearity of the mixture and a chance level of unmixing of less than 0.01%, as we have tested by numerical simulations. However, there remains a failure rate of about 30%, which is puzzling given the perfect performance on the surrogate matrices (Sec. 3.2.2). We investigate this in the next section.

3.2.4 Analysis of failure cases

Why did ISFA fail in about 30% of the cases where a good solution was available by linear regression? The values of the objective function Ψ_{ISFA} (3.3) and its two parts Ψ_{ICA} and Ψ_{SFA} give us some information about possible reasons. Consider the following four different cases:

1. In 1 out of the 35 true failures for ISFA^3 and never for ISFA^5 it is the case that Ψ_{ISFA} of the sources estimated by ISFA is greater than the Ψ_{ISFA} of the sources estimated by linear regression. In this case the algorithm obviously got stuck in a local optimum.

2. In 15 and 26 out of the 35 and 49 true failures for ISFA^3 and ISFA^5, respectively, Ψ_{ISFA} of the sources estimated by ISFA is smaller than the Ψ_{ISFA} of the sources estimated by linear regression, but either Ψ_{ICA} or Ψ_{SFA} is greater than the corresponding linear-regression value. This indicates that the tuning of the weighting factors b_{SFA} and b_{ICA} might not have been fine enough. However, it could also be that there is an abrupt transition between solutions where Ψ_{ICA} is greater to solutions where Ψ_{SFA} is greater than the corresponding linear-regression value.

3. In 6 and 3 out of the 35 and 49 true failures for ISFA^3 and ISFA^5, respectively, Ψ_{ICA} and Ψ_{SFA} of the sources estimated by ISFA are both smaller than the ones of the linear-regression estimate and greater than the ones of the original sources. Neither a local optimum nor the weighting factors are a plausible cause for the failure in these cases. It might be that the expansion was too low-dimensional and that a higher-dimensional expansion would have yielded the correct solution.

4. In 13 and 20 out of the 35 and 49 true failures for ISFA^3 and ISFA^5, respectively, Ψ_{ICA} and Ψ_{SFA} of the sources estimated by ISFA are both smaller than the ones of the original sources. In this case the solution found is even better than the original sources in terms of the objective function, which indicates that there is something wrong with the objective function.

It might be possible to eliminate the failures of the first three cases by refining the algorithm, e.g. by tuning the weighting factors better or by going to higher polynomials, but Case 4 is more fundamental and requires to reconsider the objective function itself.

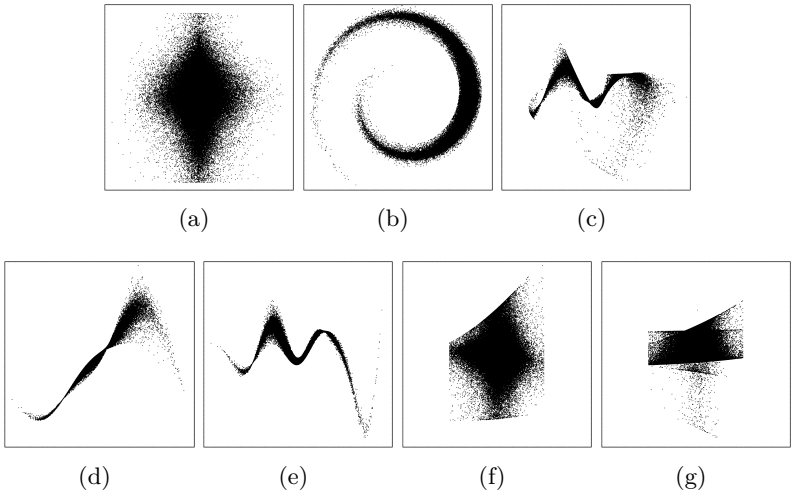


Figure 3.3: Scatter plot of two sources, their nonlinear mixture, and the sources estimated by ISFA in a failure case. (a) Sources, (b) mixture, (c) sources estimated by ISFA^3, (d) first source vs. estimated first source (corr. coeff. 0.9771), (e) first source vs. estimated second source (corr. coeff. 0.0377), (f) second source vs. estimated first source (corr. coeff. 0.0197), (g) second source vs. estimated second source (corr. coeff. 0.1301).

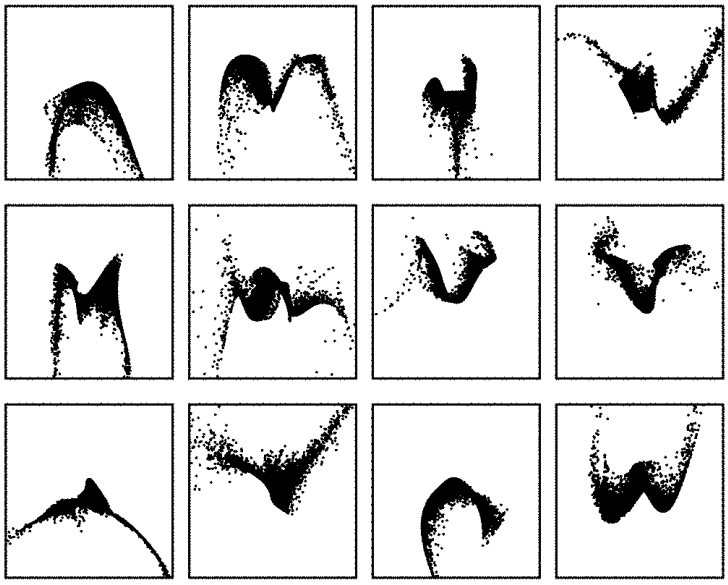
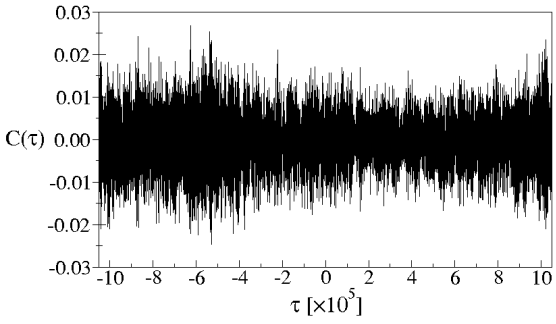
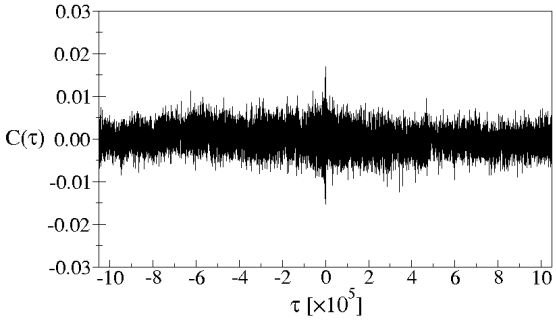


Figure 3.4: Scatter plots of the sources estimated by ISFA for some failure cases. It is clear that in these cases the signal components are not statistically independent even though the ICA-term indicates so.



(a)



(b)

Figure 3.5: Cross-correlation functions of a failure case: (a) cross-correlation function of the original sources, (b) cross-correlation function of the estimated sources. Same dataset as in Fig. 3.3.

In this latter case, the signals extracted by ISFA appear to be both slower *and* more mutually independent than the original sources. However, scatter plots of the estimated sources reveal that they are not statistically independent at all, but often one is largely a function of the other, see Figures 3.3 and 3.4. Thus the ICA part of the objective function is not strong enough to assure statistical independence of the estimated sources. The cross-correlation functions shown in Figure 3.5 indicate that this problem is not due to the specific choice of the time delays, because the time-delayed cross-correlations of the estimated sources (mean $\pm$ std $= 0 \pm 0.0028$) are overall smaller than the ones of the original sources (0 ± 0.0066). Even using different or more time delays, such dataset would have been processed incorrectly. We conclude that any measure of independence based on time-delayed correlations would be insufficient in our context.

Figure 3.5 suggested to us that sources with a large standard deviation of their cross-correlation function might be particularly difficult to separate with our ISFA algorithm. We tested this hypothesis but did not find a significant correlation with the failure cases.

For an expansion with polynomials of degree three even linear regression fails if the standard deviation of the first signal, which goes along the spiral, is large. For polynomials of degree five linear regression always worked in our examples but we suspected that separation might still be more difficult for sources with large standard deviation, but again, we did not find a significant correlation with the failure cases.

We argue here that the failures must be attributed to the weakness of the ICA-term in the objective function. If the SFA-term were too weak, it could happen that all output signal components are truly statistically independent but at least some of them are too quickly varying, so that they are not correlated to the sources but to some nonlinearly distorted version of the sources, something we did not observe. Also the success in detecting the failure cases based on higher-order cumulants (see next section) indicates that the failures are due to the ICA-term.

3.2.5 Unsupervised detection of failure cases

A failure rate of about 30% (or even up to 50% for ISFA^3 if one also counts the cases in which even linear regression was not able to recover the sources) is obviously not acceptable, unless one can detect the failure cases in an unsupervised manner. We use the weighted sum over the third and fourth order cross-cumulants,

$$\Psi_{34}(\mathbf{u}) := \frac{1}{3!} \sum_{ijk \neq iii} \left(C_{ijk}^{(\mathbf{u})}\right)^2 + \frac{1}{4!} \sum_{ijkl \neq iiii} \left(C_{ijkl}^{(\mathbf{u})}\right)^2 , \tag{3.18}$$

as an independent measure of statistical independence to indicate with high values those cases in which the second order ICA-term has failed to yield independent output signal components. The factors $\frac{1}{3!}$ and $\frac{1}{4!}$ arise from an expansion of the Kullback-Leibler divergence in $\mathbf{u}$, which provides a rigorous derivation of this criterion (Comon, 1994; McCullagh, 1987). The Receiver Operating Characteristic (ROC) curves in Figure 3.6 show that $\Psi_{34}(\mathbf{u})$ is a good measure of success. These tests also included the cases where linear regression was not able to recover the sources. The area under the ROC curves is 0.952 and 0.988 for ISFA^3 and ISFA^5, respectively.

3.3 Conclusion

In this chapter we have addressed the problem of nonlinear blind source separation. It is known that in contrast to the linear case statistical independence alone is not a sufficient criterion for separating sources from a nonlinear mixture; additional criteria are needed to solve the problem of selecting the true sources (or good representatives thereof) from the many possible output signal components that would be statistically independent of other components. We claim here that for source signals with significant autocorrelations for time delay one temporal slowness is a good criterion to solve this

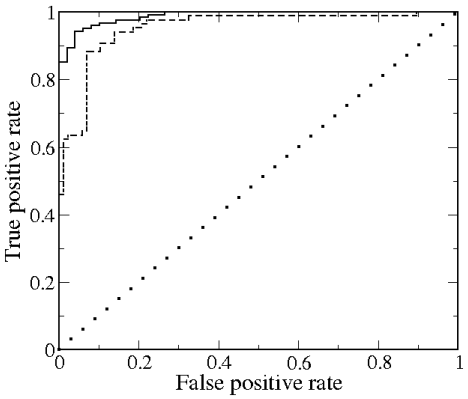


Figure 3.6: ROC curves for the test of successful source separation based on $\Psi_{34}(\mathbf{u})$, the weighted sum of third- and fourth-order cross-cumulants. The area under the curves is 0.952 and 0.988 for ISFA^3 (dashed line) and ISFA^5 (solid line), respectively.

selection problem, because the slow components are those most likely related to the true sources by an invertible transformation; non-invertible transformations would typically lead to more quickly varying components.

Based on this assumption, we have derived an objective function that combines a term from second-order Independent Component Analysis (ICA) with a term derived from Slow Feature Analysis (SFA). Optimization of the new objective function is achieved by successive Givens rotations, a method often used in context of ICA. We refer to the resulting algorithm as Independent Slow Feature Analysis (ISFA) to indicate the combination of ICA and SFA.

The algorithm is somewhat unusual in that only a small submatrix of large time-delayed correlation matrices are being diagonalized by the Givens rotations (usually the full matrices are being diagonalized). This opens the question of the uniqueness of the solution. Using randomly generated pseudo-correlation-matrices we have found that indeed a minimum number of time delays is needed to obtain unique and meaningful solutions. For instance, if the upper left 2×2-submatrix of 9×9-matrices have to be diagonalized, at least 18 such matrices are needed to obtain a meaningful solution that would be very unlikely to emerge by accident; with 17 matrices on the other hand good diagonalization can be achieved reliably even for random symmetric matrices. With (sufficiently many) surrogate matrices, structured such that they have a unique solution, we have subsequently verified that the algorithm reliably converges to the correct solution.

With tests on quite an extreme nonlinear mixture of two audio signals we have shown

that ISFA is indeed able to perform nonlinear blind source separation, often with high precision. However, in about 30% of the cases in which the true sources could have been extracted with the nonlinearity used (as verified by regression) ISFA failed to extract them. In many of these cases the extracted signals were actually better than the original sources in both the SFA-term as well as the ICA-term of the objective function. This was a surprising finding for us, since it seems to contradict our basic assumption that a combination of slowness and statistical independence should permit reliable nonlinear blind source separation. Closer inspection, however, has revealed that the extracted output signal components only appear to be statistically independent in terms of the time-delayed second-order moments but that they are often highly related, as can be seen by visual inspection (Fig. 3.4) and automatically detected with a measure $\Psi_{34}(\mathbf{u})$ based on higher-order cumulants. This is not a consequence of the particular choice of time delays we have used but would be expected for any general set of time delays, as can be seen from the cross-correlation functions (Fig. 3.5).

We believe that two important conclusions can be drawn from these results. Firstly, the success cases indicate that combining slowness and statistical independence is a promising approach to nonlinear blind source separation. Secondly, any measure of statistical independence based on (time-delayed) second-order moments is too weak to guarantee statistical independence in our context; it might even be too weak in any context where the dimensionality of the space in which the signal components are searched for is significantly larger than the number of components.

For a possible theoretical account of the failure of second-order ICA in our context consider the following example. Given a symmetrically distributed source s_1 the correlation between, for instance, s_1 and s_1^2 vanishes (Harmeling et al., 2003, sec. 4.1). To the extent that this also holds for time-shifted versions $s_1(t)$ and $s_1^2(t+\tau)$ (cf. Harmeling et al., 2003, sec. 5.4) the statistical dependence between s_1 and s_1^2 does not manifest itself in the time-delayed correlations. Thus, second-order ICA cannot be expected to prevent extraction of s_1 and s_1^2 as the estimated sources, which can easily lead to a failure case, if s_1^2 is more slowly varying than, e.g., s_2 .

A failure rate of 30% would render the algorithm useless if it were not possible to detect the failure cases. We have shown that the measure $\Psi_{34}(\mathbf{u})$, which is based on higher-order cumulants, permits failure detection with high reliability; the area under the ROC curve is greater than 0.95 resulting in a true positive rate of 90% and 94% at a false positive rate of 5% and 10% for ISFA^3 and ISFA^5, respectively.

It might be possible to use $\Psi_{34}(\mathbf{u})$ not only to detect the failure cases but also to automatically tune the weight b_{ICA} given $b_{\mathrm{SFA}} = 1$ and to determine the number of sources. For the former one could start with a small value of b_{ICA}, so that only the SFA-term is effective and the extracted components might not be independent, and then increase b_{ICA}, so that the ICA-term becomes increasingly effective, until the value of $\Psi_{34}(\mathbf{u})$ drops below a certain threshold. Similarly, for determining the number of sources one could start by running the algorithm with only two output components to

be extracted and successively increase the number of components. One would then stop if adding another component would increase $\Psi_{34}(\mathbf{u})$ significantly (which can obviously be detected only *a posteriori*).

More interesting, however, would be to use higher-order cumulants more directly to improve the algorithm. For instance, one could define a new objective function that is a combination of the SFA-term used here and an ICA-term like $\Psi_{34}(\mathbf{u})$. Given the high reliability with which $\Psi_{34}(\mathbf{u})$ can detect failure cases, we expect better performance with such a new objective function. However, higher-order cumulants are expensive to compute, especially for high-dimensional and long signals, so that there is probably a trade-off between reliability and computational complexity. Exploring these possibilities will be subject of our future research.

4 The *Slowness* Theorem

4.1 Theorem Statement

Assume a continuous input signal $x(t)$ is given, which is defined on a closed interval, and it is such that its probability density function $p_x(x)$ exists as well as the probability density function of its derivative $p_{\dot{x}}(\dot{x})$. The signal is regular enough, so that the first and second moment of the p.d.f. are defined. The p.d.f. of the derivative of the signal is required to have at least a first moment. Formally we can express our assumptions as follows:

$$\begin{aligned} & x : [t_a, t_b] \longrightarrow [x_a, x_b], \\ & x \in \mathcal{C}^0([t_a, t_b]) \\ & x \text{ such that } \exists\, \mathbf{E}(\dot{x}^2),\, \mathbf{E}(x),\, \mathbf{E}(x^2). \end{aligned} \tag{4.1}$$

The new signal $y(t)$ obtained as a result of applying a continuous function f to the signal $x(t)$ is a *transformation* of the signal. Given a transformation function $f(x)$ with similar regularity as the signal x:

$$\begin{aligned} & f : [x_a, x_b] \longrightarrow \mathbb{R}, \\ & f \in \mathcal{C}^0([x_a, x_b]) \\ & f \text{ such that } \exists\, \mathbf{E}(\dot{y}^2),\, \mathbf{E}(y),\, \mathbf{E}(y^2), \end{aligned} \tag{4.2}$$

if we call Φ the space of all functions f fulfilling the constraints in 4.2, we define the set Γ of all transformed signals:

$$\Gamma := \Big\{ y(t) = f[x(t)]\,, \forall f \in \Phi \Big\}, \quad t \in [t_a, t_b]. \tag{4.3}$$

The *slowness* of a signal can be measured by the so-called *delta value*:

$$\Delta(x) := \frac{\mathbf{E}(\dot{x}^2)}{\mathbf{E}(x^2) - \mathbf{E}(x)^2} \tag{4.4}$$

The *slowest output signal* is a signal $s(t) = f_s[x(t)]$ such that

$$\Delta(s) \leq \Delta(y) \quad \forall y \in \Gamma. \tag{4.5}$$

In the next section we are going to prove that the function f_s **is invertible on** $[x_a, x_b]$.

4.2 Proof

We sketch in the following a proof by contradiction. We show that if a non-invertible function $\tilde{f}$ generates the slowest output signal $\tilde{s}(t)$, then we can always build from $\tilde{f}$ a corresponding invertible function f_s which generates a slower signal $s(t)$ ($\Delta(s) < \Delta(\tilde{s})$).

First we introduce some nomenclature. Given a certain function f, a point x_0 in the open interval (x_a, x_b) where the function f has a local maximum or minimum is an *internal maximum* or, respectively an *internal minimum*. A formal definition reads:

- internal maximum: $\exists\, \delta_1,\, \delta_2 > 0$ such that:

$$\frac{df}{dx} \begin{cases} \geq 0 \text{ if } x \in (x_0 - \delta_1, x_0) \\ \leq 0 \text{ if } x \in (x_0, x_0 + \delta_2) \\ > 0 \text{ if } x = x_0 - \delta_1 \\ < 0 \text{ if } x = x_0 + \delta_2 \end{cases} \tag{4.6}$$

- internal minimum: $\exists\, \delta_1,\, \delta_2 > 0$ such that:

$$\frac{df}{dx} \begin{cases} \leq 0 \text{ if } x \in (x_0 - \delta_1, x_0) \\ \geq 0 \text{ if } x \in (x_0, x_0 + \delta_2) \\ < 0 \text{ if } x = x_0 - \delta_1 \\ > 0 \text{ if } x = x_0 + \delta_2 \end{cases} \tag{4.7}$$

Internal maxima and internal minima together form the set of *internal extrema* for the function f.

If a local extremum of the function f lies on the boundary of the closed interval $[x_a, x_b]$, the points x_a or x_b are an *external minimum* or, respectively, an *external maximum*. The formal definition reads:

- external minimum: $\exists\, \delta > 0$ so that:

$$\frac{df}{dx} \begin{cases} \geq 0 \text{ if } x \in (x_a, x_a + \delta) \\ > 0 \text{ if } x = x_a + \delta \end{cases} \tag{4.8}$$

or

$$\frac{df}{dx} \begin{cases} \leq 0 \text{ if } x \in (x_b - \delta, x_b) \\ < 0 \text{ if } x = x_b - \delta \end{cases} \tag{4.9}$$

- external maximum: $\exists\, \delta > 0$ so that:

$$\frac{df}{dx} \begin{cases} \leq 0 \text{ if } x \in (x_a, x_a + \delta) \\ < 0 \text{ if } x = x_a + \delta \end{cases} \tag{4.10}$$

or

$$\frac{df}{dx} \begin{cases} \geq 0 \text{ if } x \in (x_b - \delta, x_b) \\ > 0 \text{ if } x = x_b - \delta \end{cases} \tag{4.11}$$

If a local extremum is also a global one, the point x_M is a *global maximum* for f if:

$$f(x) \leq f(x_M) \quad \forall x \in [x_a, x_b]\,, \tag{4.12}$$

or, respectively, x_m is a *global minimum* for f if:

$$f(x) \geq f(x_m) \quad \forall x \in [x_a, x_b]\,. \tag{4.13}$$

Note that a global extremum must be either an internal extremum or an external extremum. Note moreover that if the function f is in Φ (see conditions in 4.2), then there always exists at least one global maximum and one global minimum for f, because the function is continuous and defined on a closed interval.

The interval (x_0, x_1) is defined to be a *plateau* if $\exists\, \delta_1, \delta_2 > 0$ such that:

$$f(x) = f_c \in \mathbb{R} \text{ if } x \in [x_0, x_1] \tag{4.14}$$

$$\frac{df}{dx} \begin{cases} > 0 \text{ if } x \in (x_0 - \delta_1, x_o) \\ = 0 \text{ if } x \in (x_0, x_1) \\ > 0 \text{ if } x \in (x_1, x_1 + \delta_2) \end{cases} \tag{4.15}$$

or

$$\frac{df}{dx} \begin{cases} < 0 \text{ if } x \in (x_0 - \delta_1, x_o) \\ = 0 \text{ if } x \in (x_0, x_1) \\ < 0 \text{ if } x \in (x_1, x_1 + \delta_2) \end{cases} \tag{4.16}$$

If $x_0 = x_a$ or $x_1 = x_b$, then $\delta 1 = 0$ or $\delta_2 = 0$ respectively.

Given the non-invertible function $\tilde{f}$ that generates the slowest output signal $\tilde{s}$, in the following we distinguish three cases:

1. $\tilde{f}$ is not monotonous and x_a and x_b are either both external maxima or both external minima
2. $\tilde{f}$ is not monotonous and x_a and x_b are either external minimum and external maximum, respectively, or external maximum and external minimum, respectively.

3. $\tilde{f}$ is monotonous.

4.2.1 Case 1

If x_a and x_b are either both external maxima or both external minima, then there exists at least one internal minimum or, respectively, one internal maximum. Let's call x_0 the smallest internal minimum or, respectively, the largest internal maximum. Note that x_0 is then a global extremum. To simplifly the notation we define $\tilde{f}_0 = \tilde{f}(x_0)$. We define a new function f_s such that

$$\frac{df_s}{dx} = \begin{cases} \dfrac{d\tilde{f}}{dx} & \text{if } x_a < x < x_0 \\ -\dfrac{d\tilde{f}}{dx} & \text{if } x_0 < x < x_b \end{cases} \tag{4.17}$$

from which it follows that

$$\begin{aligned} f_s(x) \overset{x_0<x<x_b}{=} & \tilde{f}(x_0) + \int_{x_0}^{x} \frac{df_s}{dx} dx \\ = & \tilde{f}(x_0) - \int_{x_0}^{x} \frac{d\tilde{f}}{dx} dx \\ = & \tilde{f}(x_0) - \tilde{f}(x) + \tilde{f}(x_0)\,. \end{aligned} \tag{4.18}$$

We can then globally define the function f_s to be:

$$f_s(x) = \begin{cases} \tilde{f}(x) & \text{if } x_a \le x \le x_0 \\ 2\tilde{f}_0 - \tilde{f}(x) & \text{if } x_0 < x \le x_b\,. \end{cases} \tag{4.19}$$

We can state the main result of this subsection: *The function f_s does not have an extremum in x_0, i.e. it has one extremum less than the function $\tilde{f}$.*

To show that the new signal $s = f_s(x)$ is slower than the original $\tilde{s} = \tilde{f}(x)$, we derive here the statistical moments of the new signal s:

$$\mathbf{E}(\dot{s}^2) = \mathbf{E}\left[\left(\frac{df_s}{dx}\dot{x}\right)^2\right] = \mathbf{E}\left[\left(\frac{d\tilde{f}}{dx}\right)^2 \dot{x}^2\right] = \mathbf{E}(\dot{\tilde{s}}^2)\,, \tag{4.20}$$

$$\begin{aligned}
\mathbf{E}(s) &= \int_{x_a}^{x_b} f_s p_x dx && (4.21)\\
&= \int_{x_a}^{x_0} \tilde{f} p_x dx + \int_{x_0}^{x_b} (2\tilde{f}_0 - \tilde{f}) p_x dx && (4.22)\\
&= \int_{x_a}^{x_0} \tilde{f} p_x dx + 2\int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx + \int_{x_0}^{x_b} \tilde{f} p_x dx && (4.23)\\
&= \mathbf{E}(\tilde{s}) + 2\int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx\,, && (4.24)
\end{aligned}$$

$$\begin{aligned}
\mathbf{E}(s^2) &= \int_{x_a}^{x_b} f_s^2 p_x dx && (4.25)\\
&= \int_{x_a}^{x_0} \tilde{f}^2 p_x dx + \int_{x_0}^{x_b} (2\tilde{f}_0 - \tilde{f})^2 p_x dx && (4.26)\\
&= \int_{x_a}^{x_0} \tilde{f}^2 p_x dx + \int_{x_0}^{x_b} \tilde{f}^2 p_x dx + 4\int_{x_0}^{x_b} (\tilde{f}_0^2 - \tilde{f}_0\tilde{f}) p_x dx && (4.27)\\
&= \mathbf{E}(\tilde{s}^2) + 4\tilde{f}_0 \int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx\,. && (4.28)
\end{aligned}$$

The variance of the new signal reads:

$$\begin{aligned}
\mathbf{E}(s^2) - \mathbf{E}(s)^2 &= \mathbf{E}(\tilde{s}^2) + 4\tilde{f}_0 \int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx \\
&\quad - \left(\mathbf{E}(\tilde{s}) + 2\int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx\right)^2 \\
&= \mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2 + 4\left(\int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx\right) \\
&\quad \cdot \left(\tilde{f}_0 - \int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx - \mathbf{E}(\tilde{s})\right) \\
&= \mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2 + 4\left(\int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx\right) \cdot I_1\,. \qquad (4.29)
\end{aligned}$$

Note that:

$$\begin{aligned}
I_1 &= \tilde{f}_0 - \int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx - \mathbf{E}(\tilde{s}) \\
&= \tilde{f}_0 \int_{x_a}^{x_b} p_x dx - \int_{x_a}^{x_b} \tilde{f} p_x dx - \int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx \\
&= \int_{x_a}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx - \int_{x_0}^{x_b} (\tilde{f}_0 - \tilde{f}) p_x dx \\
&= \int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx\,. \qquad (4.30)
\end{aligned}$$

Finally we can rewrite the variance of the new signal as follows:

$$\mathbf{E}(s^2) - \mathbf{E}(s)^2 = \mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2 + 4\left(\int_{x_a}^{x_0}(\tilde{f}_0 - \tilde{f})p_x dx\right)\left(\int_{x_0}^{x_b}(\tilde{f}_0 - \tilde{f})p_x dx\right). \quad (4.31)$$

Note that x_0 is a global extremum and, because of Eqs. (4.12, 4.13) the two integrals in Eq. (4.31) are either both positive or both negative. Thus:

$$\begin{aligned}\frac{\Delta(\tilde{s})}{\Delta(s)} &= \frac{\mathbf{E}(\dot{\tilde{s}}^2)}{\mathbf{E}(\dot{s}^2)} \cdot \frac{\mathbf{E}(s^2) - \mathbf{E}(s)^2}{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2} \\ &= \frac{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2 + K}{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2} \\ &= 1 + \frac{K}{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2}\end{aligned} \quad (4.32)$$

where $K \geq 0$. This impliest that signal s is slower than $\tilde{s}$:

$$\Delta(\tilde{s}) \geq \Delta(s) \quad (4.33)$$

the above procedure is graphically depicted in subplot 1 of Figure 4.1.

4.2.2 Case 2

If x_a and x_b are either external minimum and external maximum, respectively, or external maximum and external minimum, respectively, then the function $\tilde{f}$ must have either zero or an even number of internal extrema, half of which are internal maxima and half of which are internal minima. If $\tilde{f}$ has no internal extrema, it is already monotonous and we can skip the rest. On the other hand, if for example x_a is an external minimum and x_b an external maximum, it is easy to see that the internal extremum closest to x_a must be a maximum, followed by a minimum, and so on. Define x_0 to be the largest of the internal maxima, and x_1 the smallest of the internal minima found in the interval (x_0, x_b). Note that there is always at least one of such minima. We have:

$$\tilde{f}(x) \leq \tilde{f}_0 \text{ for } x \in (x_a, x_1)\,, \quad (4.34)$$

$$\tilde{f}(x) \geq \tilde{f}_1 \text{ for } x \in (x_0, x_b)\,, \quad (4.35)$$

$$\tilde{f}_0 > \tilde{f}_1\,, \quad (4.36)$$

where $\tilde{f}_0 = \tilde{f}(x_0)$ and $\tilde{f}_1 = \tilde{f}(x_1)$.

The only other possibility is that x_a is an external maximum and x_b an external minimum. In analogy with the previous case, we choose x_0 to be the smallest of the internal minima, and x_1 to be the largest of the internal maxima found in the interval

(x_0, x_b). Then we have:

$$\tilde{f}(x) \geq \tilde{f}_0 \text{ for } x \in (x_a, x_1) \tag{4.37}$$
$$\tilde{f}(x) \leq \tilde{f}_1 \text{ for } x \in (x_0, x_b) \tag{4.38}$$
$$\tilde{f}_0 < \tilde{f}_1 \,. \tag{4.39}$$

We define a new function f_s such that:

$$\frac{df_s}{dx} = \begin{cases} -\dfrac{d\tilde{f}}{dx} & \text{if } x_a \leq x < x_0 \text{ or } x_1 < x \leq x_b \\ \dfrac{df}{dx} & \text{if } x_0 < x < x_1 \end{cases} \tag{4.40}$$

The new function definition then reads:

$$f_s(x) = \begin{cases} 2\tilde{f}0 - \tilde{f}(x) & \text{if } x_a \leq x < x_0 \\ \tilde{f}(x) & \text{if } x_0 \leq x \leq x_1 \\ 2\tilde{f}_1 - \tilde{f}(x) & \text{if } x_1 < x \leq x_b \end{cases} \tag{4.41}$$

The function f_s does not have internal extrema in x_0 and x_1, i.e. f_s has two extrema less than $\tilde{f}$. We derive the statistical moments of the new signal $s = f_s(x)$:

$$\mathbf{E}(\dot{s}^2) = \mathbf{E}\left[\left(\frac{df_s}{dx}\dot{x}\right)^2\right] = \mathbf{E}\left[\left(\frac{d\tilde{f}}{dx}\right)^2 \dot{x}^2\right] = \mathbf{E}(\dot{\tilde{s}}^2)\,, \tag{4.42}$$

$$\mathbf{E}(s) = \int_{x_a}^{x_b} f_s p_x dx \tag{4.43}$$
$$= \int_{x_a}^{x_0} (2\tilde{f}_0 - \tilde{f}) p_x dx + \int_{x_0}^{x_1} \tilde{f} p_x dx + \int_{x_1}^{x_b} (2\tilde{f}_1 - \tilde{f}) p_x dx \tag{4.44}$$
$$= 2\int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx + \int_{x_a}^{x_0} \tilde{f} p_x dx + \int_{x_0}^{x_1} \tilde{f} p_x dx \tag{4.45}$$
$$+ 2\int_{x_1}^{x_b} (\tilde{f}_1 - \tilde{f}) p_x dx + \int_{x_1}^{x_b} \tilde{f} p_x dx \tag{4.46}$$
$$= \mathbf{E}(\tilde{s}) + 2\int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx + 2\int_{x_1}^{x_b} (\tilde{f}_1 - \tilde{f}) p_x dx \tag{4.47}$$
$$= \mathbf{E}(\tilde{s}) + 2I_1 + 2I_2\,, \tag{4.48}$$

$$\mathbf{E}(s^2) = \int_{x_a}^{x_b} f_s^2 p_x dx \tag{4.49}$$
$$= \int_{x_a}^{x_0} (2\tilde{f}_0 - \tilde{f})^2 p_x dx + \int_{x_0}^{x_1} \tilde{f}^2 p_x dx + \int_{x_1}^{x_b} (2\tilde{f}_1 - \tilde{f})^2 p_x dx \tag{4.50}$$
$$= \int_{x_a}^{x_0} (4\tilde{f}_0^2 + \tilde{f}^2 - 4\tilde{f}_0\tilde{f}) p_x dx + \int_{x_0}^{x_1} \tilde{f}^2 p_x dx \tag{4.51}$$
$$+ \int_{x_1}^{x_b} (4\tilde{f}_1^2 + \tilde{f}^2 - 4\tilde{f}_1\tilde{f}) p_x dx \tag{4.52}$$
$$= \mathbf{E}(\tilde{s}^2) + 4\tilde{f}_0 \int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx + 4\tilde{f}_1 \int_{x_1}^{x_b} (\tilde{f}_1 - \tilde{f}) p_x dx \tag{4.53}$$
$$= \mathbf{E}(\tilde{s}^2) + 4\tilde{f}_0 I_1 + 4\tilde{f}_1 I_2 \,. \tag{4.54}$$

We can now calculate the variance of the new signal:

$$\begin{aligned}
\mathbf{E}(s^2) - \mathbf{E}(s)^2 &= \mathbf{E}(\tilde{s}^2) + 4\tilde{f}_0 I_1 + 4\tilde{f}_1 I_2 - (\mathbf{E}(\tilde{s}) + 2(I_1 + I_2))^2 \\
&= \mathbf{E}(\tilde{s}^2) - (\mathbf{E}(\tilde{s}))^2 + 4\tilde{f}_0 I_1 \\
&\quad + 4\tilde{f}_1 I_2 - 4(I_1 + I_2)^2 - 4\mathbf{E}(\tilde{s})(I_1 + I_2) \\
&= \mathbf{E}(\tilde{s}^2) - (\mathbf{E}(\tilde{s}))^2 + 4I_1(\tilde{f}_0 - I_1 - \mathbf{E}(\tilde{s}) - I_2) \\
&\quad + 4I_2(\tilde{f}_1 - I_2 - \mathbf{E}(\tilde{s}) - I_1) \\
&= \mathbf{E}(\tilde{s}^2) - (\mathbf{E}(\tilde{s}))^2 + 4I_1 I_3 + 4I_2 I_4 \,.
\end{aligned} \tag{4.55}$$

Note that:

$$\begin{aligned}
I_3 &= \tilde{f}_0 - I_1 - \mathbf{E}(\tilde{s}) - I_2 \\
&= \tilde{f}_0 \int_{x_a}^{x_b} p_x dx - \int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx - \int_{x_a}^{x_b} \tilde{f} p_x dx - \int_{x_1}^{x_b} (\tilde{f}_1 - \tilde{f}) p_x dx \\
&= \int_{x_0}^{x_1} (\tilde{f}_0 - \tilde{f}) p_x dx + (\tilde{f}_0 - \tilde{f}_1) \int_{x_1}^{x_b} p_x dx \,,
\end{aligned} \tag{4.56}$$

$$\begin{aligned}
I_2 &= \tilde{f}_1 - I_2 - \mathbf{E}(\tilde{s}) - I_1 \\
&= \tilde{f}_1 \int_{x_a}^{x_b} p_x dx - \int_{x_1}^{x_b} (\tilde{f}_1 - \tilde{f}) p_x dx - \int_{x_a}^{x_b} \tilde{f} p_x dx - \int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx \\
&= \int_{x_0}^{x_1} (\tilde{f}_1 - \tilde{f}) p_x dx + (\tilde{f}_1 - \tilde{f}_0) \int_{x_a}^{x_0} p_x dx \,.
\end{aligned} \tag{4.57}$$

Because of our definition for x_0 and x_1, only the following combinations of signs are possible:

- if x_a is an external minimum and x_b an external maximum, because of Eqs. (4.34-4.36), we have:

$$I_1 \geq 0, \quad I_3 \geq 0, \quad I_2 \leq 0, \quad I_4 \leq 0 \tag{4.58}$$

- if x_a is an external maximum and x_b an external minimum, because of Eqs. (4.37-4.39), we have:

$$I_1 \leq 0, \quad I_3 \leq 0, \quad I_2 \geq 0, \quad I_4 \geq 0 \tag{4.59}$$

In both cases the term $4I_1I_3 + 4I_2I_4$ in Eq. (4.55) is positive. Thus:

$$\begin{aligned} \frac{\Delta(\tilde{s})}{\Delta(s)} &= \frac{\mathbf{E}(\dot{\tilde{s}}^2)}{\mathbf{E}(\dot{s}^2)} \cdot \frac{\mathbf{E}(s^2) - \mathbf{E}(s)^2}{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2} \\ &= \frac{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2 + K}{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2} \\ &= 1 + \frac{K}{\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2} \end{aligned} \tag{4.60}$$

where $K \geq 0$. The signal s is then slower than $\tilde{s}$:

$$\Delta(\tilde{s}) \geq \Delta(s)\,. \tag{4.61}$$

An example of the above procedure is shown in subplot 2 of Figure 4.1.

4.2.3 Iterative elimination of local extrema

We have shown a systematic way to eliminate internal extrema from a non-invertible function. Because the functions in Γ are continuous and piecewise differentiable there is always a finite number of such extrema, that can be eliminated iteratively applying the previous methods. The resulting function has then no internal extrema, i.e. it is monotonous. We are then left with a function that could still be non-invertible, because there could be *plateaus*, where the function is constant. We consider this case in the next section.

4.2.4 Case 3

If $\tilde{f}$ is monotonous and it has at least one plateau in (x_0, x_1), we define a new function f_s such that

$$\frac{df_s}{dx} = \begin{cases} \dfrac{d\tilde{f}}{dx} & \text{if } x_a < x < x_0 \text{ or } x_1 < x < x_b \\ \varepsilon & \text{if } x_0 < x < x_1\,. \end{cases} \tag{4.62}$$

The function f_s can be globally defined as

$$f_s(x) = \begin{cases} \tilde{f}(x) & \text{if } x_a \leq x < x_0 \\ \tilde{f}(x) + \varepsilon(x - x_0) & \text{if } x_0 \leq x \leq x_1 \\ \tilde{f}(x) + \varepsilon(x_1 - x_0) & \text{if } x_1 < x \leq x_b\,, \end{cases} \tag{4.63}$$

where

$$|\varepsilon| \ll 1\,, \tag{4.64}$$

$$\varepsilon > 0 \text{ if } \tilde{f} \text{ is increasing}\,, \tag{4.65}$$

$$\varepsilon < 0 \text{ if } \tilde{f} \text{ is decreasing}\,. \tag{4.66}$$

The function f_s does not have a plateau in the interval (x_0, x_1), and this has been substituted by a ramp of slope ε. We calculate the statistical moments of the new signal $s = f_s(x)$:

$$\begin{aligned}
\mathbf{E}(\dot{s}^2) &= \mathbf{E}\left[\left(\frac{df_s}{dx}\dot{s}\right)^2\right] \\
&= \int_{x_a}^{x_0}\int_{\dot{x}_a}^{\dot{x}_b}\left(\frac{d\tilde{f}}{dx}\right)^2 \dot{x}^2 p_{x,\dot{x}}(x,\dot{x})d\dot{x}dx \\
&\quad + \int_{x_1}^{x_b}\int_{\dot{x}_a}^{\dot{x}_b}\left(\frac{df_s}{dx}\right)^2 \dot{x}^2 p_{x,\dot{x}}(x,\dot{x})d\dot{x}dx + \varepsilon^2\int_{x_0}^{x_1}\int_{\dot{x}_a}^{\dot{x}_b}\dot{x}^2 p_{x,\dot{x}}(x,\dot{x})d\dot{x}dx \\
&= \mathbf{E}[\dot{\tilde{s}}^2] + K_1\varepsilon^2
\end{aligned} \tag{4.67}$$

with:

$$K_1 = \int_{x_0}^{x_1}\int_{\dot{x}_a}^{\dot{x}_b}\dot{x}^2 p_{x,\dot{x}}(x,\dot{x})d\dot{x}dx \geq 0\,, \tag{4.68}$$

$$\begin{aligned}
\mathbf{E}(s) &= \int_{x_a}^{x_b} f_s p_x dx \\
&= \int_{x_a}^{x_0}\tilde{f}p_x dx + \int_{x_0}^{x_1}(\tilde{f} + \varepsilon(x - x_0))p_x dx + \int_{x_1}^{x_b}(\tilde{f} + \varepsilon(x_1 - x_0))p_x dx \\
&= \mathbf{E}(\tilde{s}) + \varepsilon(I_1 + (x_1 - x_0)I_2)
\end{aligned} \tag{4.69}$$

with

$$I_1 = \int_{x_0}^{x_1}(x - x_0)p_x dx \geq 0 \tag{4.70}$$

$$\text{and} \quad I_2 = (x_1 - x_0)\int_{x_1}^{x_b} p_x dx \geq 0\,, \tag{4.71}$$

$$\begin{aligned}
\mathbf{E}(s^2) &= \int_{x_a}^{x_b} f_s^2 p_x dx \\
&= \int_{x_a}^{x_0} \tilde{f}^2 p_x dx + \int_{x_0}^{x_1} \tilde{f}^2 p_x dx + 2\varepsilon \int_{x_0}^{x_1} (x - x_0)\tilde{f} p_x dx \\
&\quad + \int_{x_1}^{x_b} \tilde{f}^2 p_x dx + 2\varepsilon \int_{x_1}^{x_b} (x_1 - x_0)\tilde{f} p_x dx + O(\varepsilon^2) \\
&= \mathbf{E}(\tilde{s}^2) + 2\varepsilon \left[\tilde{f}_0 I_1 + (x_1 - x_0) \int_{x_1}^{x_b} (\tilde{f} - \tilde{f}_0 + \tilde{f}_0) p_x dx \right] + O(\varepsilon^2) \\
&= \mathbf{E}(\tilde{s}^2) + 2\varepsilon \left[\tilde{f}_0 I_1 + \tilde{f}_0 (x_1 - x_0) I_2 + (x_1 - x_0) I_3 \right] + O(\varepsilon^2)
\end{aligned} \tag{4.72}$$

with

$$I_3 = \int_{x_1}^{x_b} (\tilde{f} - \tilde{f}_0) p_x dx \,. \tag{4.73}$$

Note that

$$I_3 \geq 0 \text{ if } \tilde{f} \text{ is increasing}\,, \tag{4.74}$$

$$I_3 \leq 0 \text{ if } \tilde{f} \text{ is decreasing}\,. \tag{4.75}$$

Deriving the variance of the new signal yields

$$\begin{aligned}
\mathbf{E}(s^2) - \mathbf{E}(s)^2 &= \mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2 + 2\varepsilon \Big[\tilde{f}_0 I_1 + \tilde{f}_0 (x_1 - x_0) I_2 \\
&\quad + (x_1 - x_0) I_3 - \mathbf{E}(\tilde{s})(I_1 + (x_1 - x_0) I_2) \Big] + O(\varepsilon^2) \\
&= \mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2 + \varepsilon K_2 + O(\varepsilon^2)\,,
\end{aligned} \tag{4.76}$$

where

$$\frac{K_2}{2} = \tilde{f}_0 I_1 + \tilde{f}_0 (x_1 - x_0) I_2 + (x_1 - x_0) I_3 - \mathbf{E}(\tilde{s})(I_1 + (x_1 - x_0) I_2)\,. \tag{4.77}$$

Note that

$$\begin{aligned}
\mathbf{E}(\tilde{s}) &= \int_{x_a}^{x_b} \tilde{f} p_x dx = \int_{x_a}^{x_b} (\tilde{f} - \tilde{f}_0 + \tilde{f}_0) p_x dx = \tilde{f}_0 + \int_{x_a}^{x_b} (\tilde{f} - \tilde{f}_0) p_x dx \\
&= \tilde{f}_0 + \int_{x_a}^{x_0} (\tilde{f} - \tilde{f}_0) p_x dx + \int_{x_1}^{x_b} (\tilde{f} - \tilde{f}_0) p_x dx \\
&= \tilde{f}_0 - \int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx + I_3 \\
&= \tilde{f}_0 - I_4 + I_3
\end{aligned} \tag{4.78}$$

where

$$I_4 = \int_{x_a}^{x_0} (\tilde{f}_0 - \tilde{f}) p_x dx \,. \tag{4.79}$$

and

$$I_4 \geq 0 \text{ if } \tilde{f} \text{ is increasing}\,, \tag{4.80}$$
$$I_4 \leq 0 \text{ if } \tilde{f} \text{ is decreasing}\,. \tag{4.81}$$

Then

$$\begin{aligned}
\frac{K_2}{2} &= \tilde{f}_0(I_1 + (x_1 - x_0)I_2) - (\tilde{f}_0 - I_4 + I_3)(I_1 + (x_1 - x_0)I_2) + (x_1 - x_0)I_3 \\
&= I_4(I_1 + (x_1 - x_0)I_2) + I_3((x_1 - x_0) - I_1 + (x_1 - x_0)I_2) \\
&= I_4(I_1 + (x_1 - x_0)I_2) + I_3 \Bigg((x_1 - x_0) \int_{x_a}^{x_b} p_x dx \\
&\quad - \int_{x_0}^{x_1} (x - x_0) p_x dx - (x_1 - x_0) \int_{x_1}^{x_b} p_x dx \Bigg) \\
&= I_4(I_1 + (x_1 - x_0)I_2) + I_3 \Bigg((x_1 - x_0) \int_{x_a}^{x_0} p_x dx \\
&\quad + \int_{x_0}^{x_1} (x_1 - x_0) p_x dx - \int_{x_0}^{x_1} (x - x_0) p_x dx \Bigg) \\
&= I_4(I_1 + (x_1 - x_0)I_2) + I_3 \Bigg((x_1 - x_0) \int_{x_a}^{x_0} p_x dx + \int_{x_0}^{x_1} (x_1 - x) p_x dx \Bigg) \\
&= I_4(I_1 + (x_1 - x_0)I_2) + I_3 I_5 \,,
\end{aligned} \tag{4.82}$$

where

$$I_5 = (x_1 - x_0) \int_{x_a}^{x_0} p_x dx + \int_{x_0}^{x_1} (x_1 - x) p_x dx \geq 0 \,. \tag{4.83}$$

Note that

$$K_2 \geq 0 \text{ if } \tilde{f} \text{ is increasing [Eqs. (4.70, 4.71, 4.74, 4.80, 4.83)]}\,, \tag{4.84}$$
$$K_2 \leq 0 \text{ if } \tilde{f} \text{ is decreasing [Eqs. (4.70, 4.71, 4.75, 4.81, 4.83)]}\,, \tag{4.85}$$
$$\varepsilon K_2 \geq 0 \text{ always [Eqs. (4.84, 4.65, 4.85, 4.66)]}\,. \tag{4.86}$$

We can now calculate the slowness of the new signal s:

$$A = \mathbf{E}(\dot{\tilde{s}}^2)\,, \tag{4.87}$$

$$B = \mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2\,, \tag{4.88}$$

$$\begin{aligned}\Delta(s) &= \frac{A + \varepsilon^2 K_1}{B + \varepsilon K_2} \\ &= \left(\frac{A}{B} + \varepsilon^2 \frac{K_1}{B}\right)\left(\frac{1}{1 + \varepsilon\frac{k_2}{B} + O(\varepsilon^2)}\right) \\ &= \left(\frac{A}{B} + \varepsilon^2 \frac{K_1}{B}\right)\left(1 - \varepsilon\frac{K_2}{B} + O(\varepsilon^2)\right) \\ &= \frac{A}{B} - \varepsilon K_2 \frac{A}{B^2} + O(\varepsilon^2) \\ &= \Delta(\tilde{s}) - K + O(\varepsilon^2)\end{aligned} \tag{4.89}$$

where

$$K = \varepsilon K_2 \frac{\mathbf{E}(\dot{\tilde{s}}^2)}{(\mathbf{E}(\tilde{s}^2) - \mathbf{E}(\tilde{s})^2)^2} \geq 0 \tag{4.90}$$

and where we have used the Taylor expansion:

$$\frac{1}{1+ax} = 1 - ax + O(x^2),\ \text{if } |ax| < 1 \tag{4.91}$$

with $a = \frac{K_2}{B}$ and $x = \varepsilon$.

The signal s is slower than $\tilde{s}$:

$$\Delta(\tilde{s}) > \Delta(s)\,. \tag{4.92}$$

Using the procedure depicted above and shown in subplot 3 of Figure 4.1, we can iteratively eliminate all the *plateaus* of a general non-invertible and monotonous function $\tilde{f}$. We are thus left with a function f_s that generates a signal s that is slower than $\tilde{s}$, with f_s strictly monotonous, i.e. invertible on $[x_a, x_b]$. This contradicts the starting assumption of $\tilde{f}$ as being the function generating the slowest signal $\tilde{s}$, which finally proves the theorem: The *slowest signal* $s(t) = f_s[x(t)]$ among all signals generated applying a sufficiently well-behaved function to the original signal $x(t)$ is an invertible function of x.

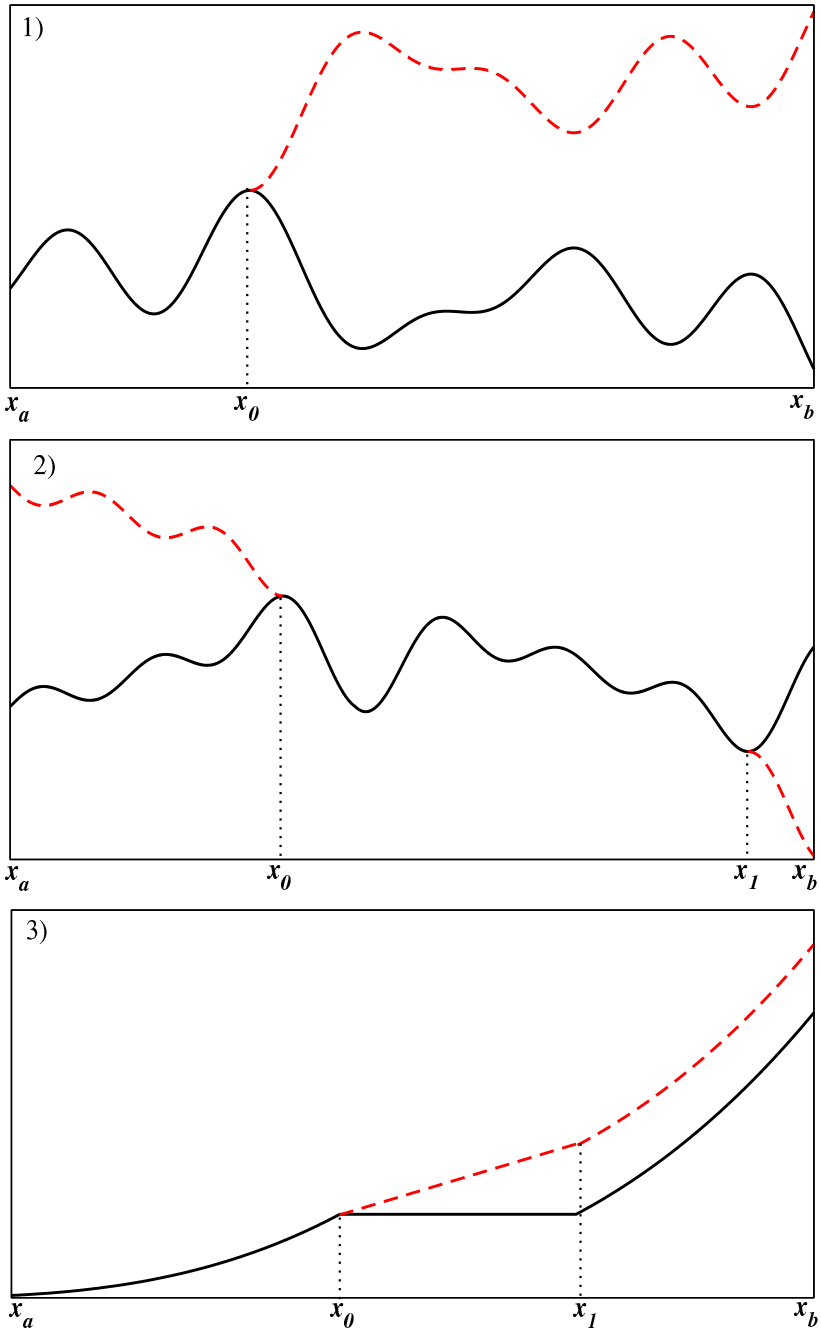


Figure 4.1: Construction of invertible slow functions: original function $\tilde{f}$ (full line) and slower function f_s (dashed line) for the three possible cases.

5 Slowness as a computational principle of the auditory cortex

Berkes and Wiskott (2005) used temporal slowness as a computational principle for the learning of receptive fields in a virtually parameter-free and unsupervised model of the primary visual cortex (V1). The learned units show a remarkable quantitative and qualitative match with the properties of complex cells in V1. The input-output function were trained on natural scene sequences and exhibit direction selectivity, non-orthogonal inhibition, end-inhibition, and side-inhibition. Furthermore, by complementing slowness with sparseness Franzius et al. (2007b) could model the self-organized formation of place cells, head-direction cells and spatial-view cells in the hippocampus. An extension of the model in Franzius et al. (2007b) can learn invariant object recognition and extract identity, position, and rotation angles of simple objects (see Franzius, Wilbert, and Wiskott, 2008b). Sprekeler et al. (2007) have shown that slowness is a plausible objective for spike-timing-dependent plasticity, which could be one possible implementation of a learning rule for slowness in spiking neurons.

These noteworthy results in the visual domain indicate that the slowness principle can explain the self-organization of cortical areas. It is therefore natural to think that similar results could be obtained in the auditory domain. An attempt to do so and the limitations thereof are documented in this chapter.

First a short overview of the auditory pathway in humans is given. Then some common representations of audio signals used in signal processing is sketched. Finally, a model of the primary auditory cortex based on the slowness principle is discussed.

5.1 The Auditory Pathway

The auditory and the visual cortex comprise a similar number of neurons. The number of peripheral receptor cells are, however, quite different. The retina is constituted of 130 million photo receptors, which through one million visual nerve fibers stimulate 100 million neurons in the primary visual cortex. The 15000 hair cells in the cochlear basilar membrane are attached to 15000 auditory nerve fibers, which project into a complex pathway within the brainstem through many nuclei up to the roughly 100 million neurons in the auditory cortex. This striking difference is due to the different time scale, resolution, and dimensionality of the incoming stimuli: The visual world is three dimensional and the receptors in the retina mirror this complexity, whereas the

auditory world gets compressed in two one dimensional signals, namely the air pressure wave, which gets perceived as sound. The receptors in the inner ear need to decode from a one dimensional signal an information flow which inherently spans several orders of magnitude in time, amplitude, and frequency.

The auditory pathway comprises four main stages: the ear (outer, middle and inner), the auditory nerve, several relay stations in the midbrain, and the auditory cortex.

5.1.1 Outer, Middle, and Inner Ear

The sound waves enter the external ear and reach the tympanum, which vibrates and transmits those vibrations to the three small bones located within the middle ear (malleus, incus, and stapes), see Figure 5.1 (top panel). Those vibrations get through another series of transmission channels (the oval window, the perilymph, the vestibular membrane) to finally hit the endolymph fluid within the cochlear duct. The incoming waves are amplified around 20 times when they hit the cochlear duct. The mechanical force transmitted from the middle ear is transformed to hydraulic pressure. This pressure induces oscillations in the basilar membrane which travel from the pit to the end of the cochlea. High frequency sounds induce higher oscillations at the base of the cochlea, where the duct is thick and narrow, whereas low frequency sounds create higher oscillations at the apex of the cochlea, where the duct is wide and thin (Figure 5.1, bottom panel). This *tonotopic* distribution is due to the physical properties of the membrane and its resonant frequencies.

5.1.2 Auditory Nerve

The mechanical-to-neural transduction happens within the organ of Corti, which describes the complex of hair cells, membranes, lamina and fluids attached to the basilar membrane, see Figure 5.2(a). The hair cells, which are stretched between the basilar and the tectorial membrane, bend according to the basilar membrane vibrations. This bending modulates the opening and closing of ion channels in the hair cells. The resulting generator potential induces neuro-transmitter release from the cells which excites the afferent nerve. The potential is excitatory or inhibitory depending on the direction of bending. Each auditory nerve fiber has a so-called characteristic frequency (CF), which is the frequency of an incoming pure tone at which least energy is needed to stimulate it. The CF of a nerve fiber is roughly the resonant frequency of oscillation of the corresponding tract of the basilar membrane. Auditory nerve fibers are typically characterized by their CF and by their Frequency threshold curves (FTC), which are a plot of the minimum intensity of a pure tone at a particular frequency needed to stimulate a particular nerve fiber just above spontaneous activity, an example is shown in Figure 5.2(b).

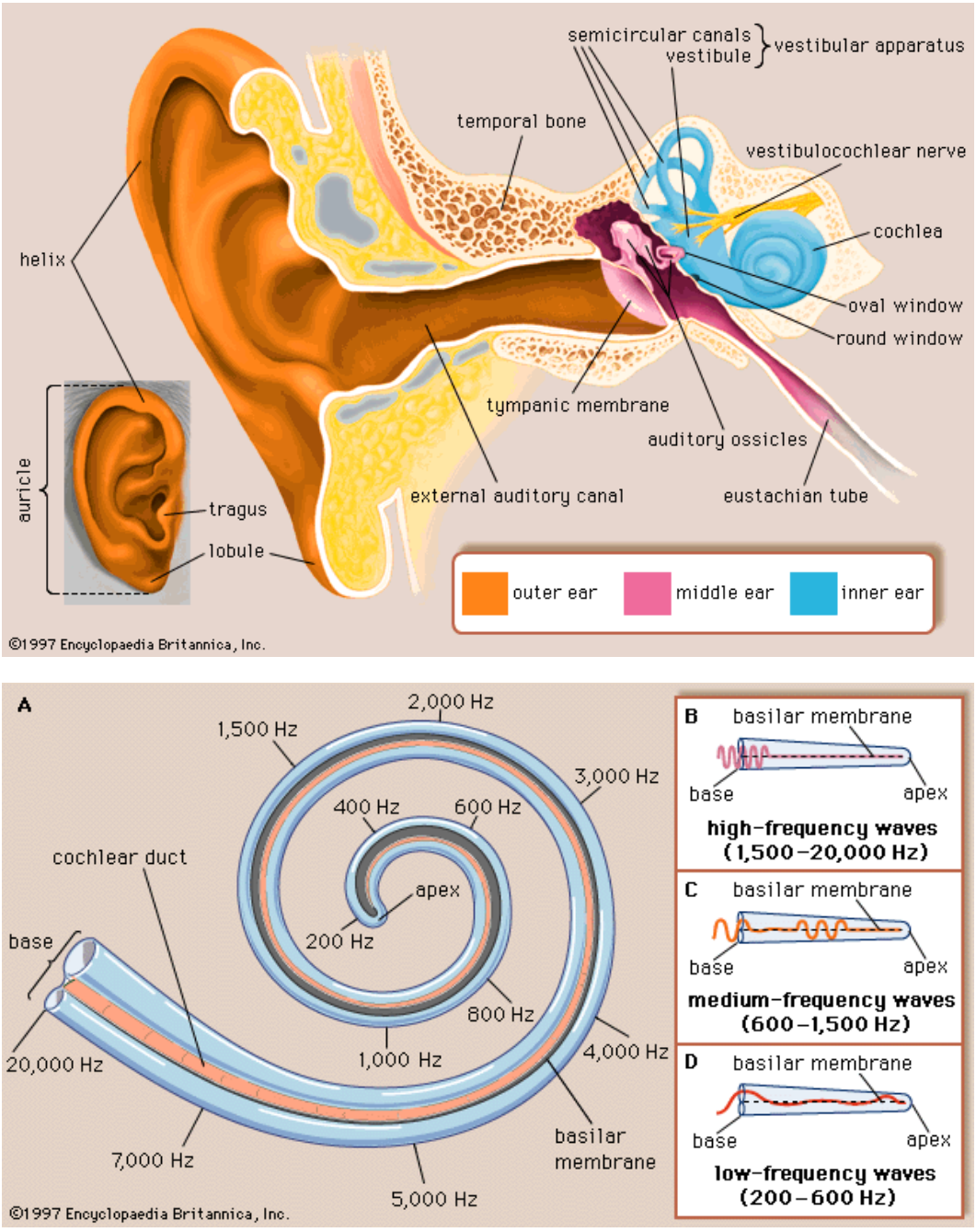
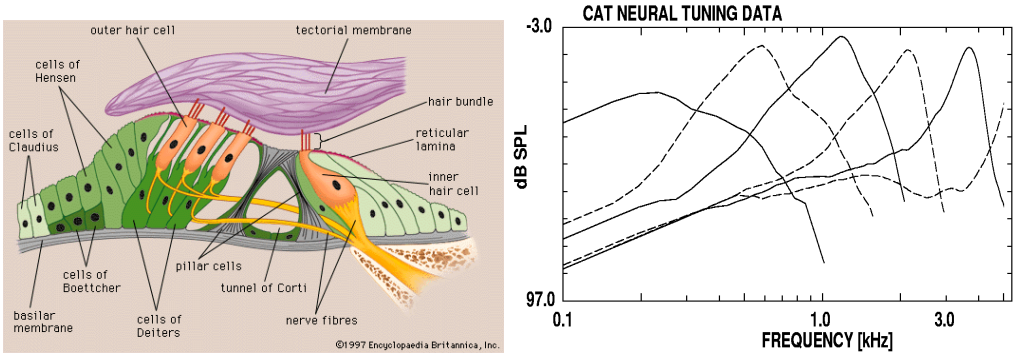


Figure 5.1: Outer, middle, inner ear and basilar membrane (from Encyclopaedia Britannica Online: `http://is.gd/g6ZGB` and `http://is.gd/g6Zvk`).



(a) (b)

Figure 5.2: a) Organ of Corti (from Encyclopaedia Britannica Online: `http://is.gd/g70r1`); b) Frequency-threshold curves for auditory nerve fibers in the cat (from Allen, 2001). Note the reversed amplitude axis.

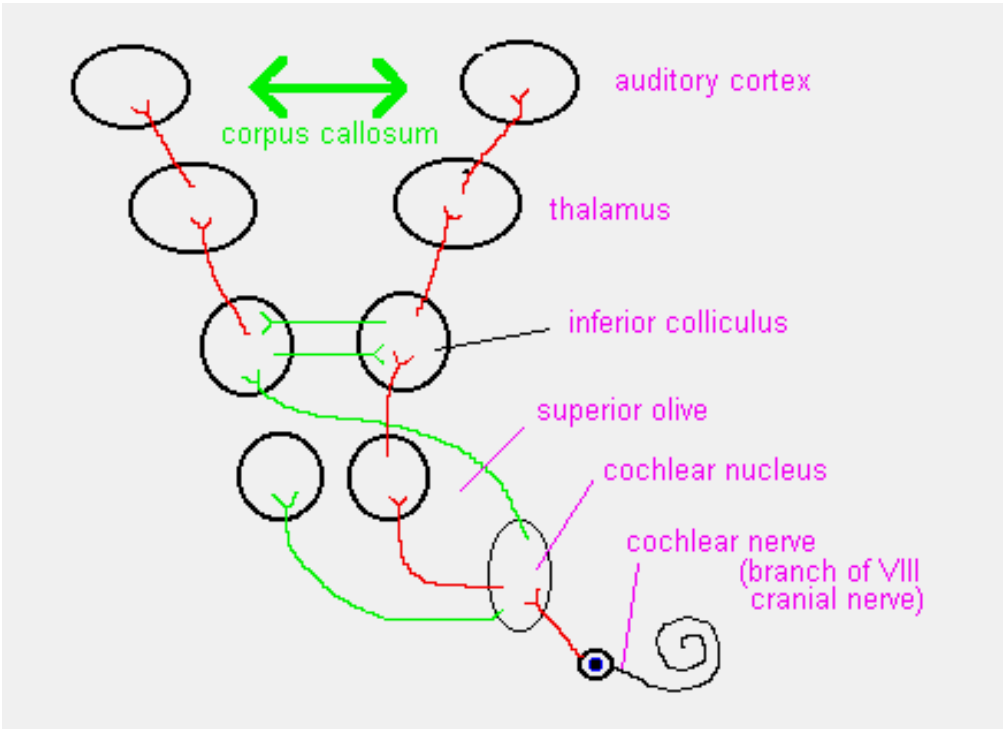


Figure 5.3: The auditory pathway through the midbrain (by Tim Jacob, `http://is.gd/g9Wnr`).

5.1.3 The Midbrain and the Auditory Cortex

Signals coming from the the cochlear nucleus are split in at least two streams and are relayed through superior olive, inferior colliculus, medial geniculate nucleus of the thalamus, with various back and lateral projections up to the auditory cortex (Figure 5.3). Neurons in these regions show several response types, which may be roughly classified in onset, offset, on-off, sustained, pauser, chopper, tonic, inhibitory. Functionally these neurons are responsible for the spectral analysis (both pitch and loudness) and for sound localization. The neurons are organized tonotopically (just like visual neurons are organized retinotopically) and their response is phase-locked to the incoming stimuli, at least for frequencies below 4 kHz.

Neurons in the auditory cortex are typically probed with pure tones or white noise. Most of the neurons in the auditory cortex do not respond to simple tones. Acoustic signals in natural environment are highly overlapping in spectral domain and subsequently in their peripheral neural representation. It is fair to say that no general consensus exists in the field about the function and organization of neurons in the primary auditory cortex. One reason may also be that it is not straightforward to define what exactly an *auditory object* may be (see for example Griffiths and Warren, 2004).

The concept of spectro-temporal receptive field (STRF) (Aertsen and Johannesma, 1981) has been used to characterize the response of neurons in the auditory cortex in a way that allows for a richer description than that offered by frequency-threshold curves and that resembles somehow the well understood receptive fields for neurons in the visual cortex (Hartline, 1940). The STRF for a neuron shows the temporal succession of acoustical stimuli which elicits the maximum response for that neuron. It is assumed that the response of the neuron to a stimulus which has not been used to estimate the STRF, can be predicted solely based on linear inter- or extrapolation from the STRF. In other words, the STRF is the best linear model that transforms the incoming signal into a prediction of the firing rate of the neuron.

Spectro-temporal receptive fields can be estimated by simple reverse correlation techniques (Boer and Kuyper, 1968) on the response to uniformly distributed white noise stimuli before each spike. However, the stimuli used to probe the neurons for STRF estimation are preferably sampled from ensembles of natural sounds, which are more relevant to the probed animal and are more likely to drive the neurons into their full dynamic range. The use of non-white noise stimuli complicates the mathematics behind the estimation of the STRF considerably, but it can be shown (Theunissen et al., 2000) that such estimation is possible and the predictions thus obtained are still relevant even for non-linear neurons. In Figure 5.4 a comparison is shown of STRFs for neurons in the avian auditory forebrain obtained by stimulating with white noise and with species behaviorally relevant stimuli.

Because of its similarities with the receptive field in the visual domain, the spectro-temporal receptive field represent an interesting objective for testing the slowness hy-

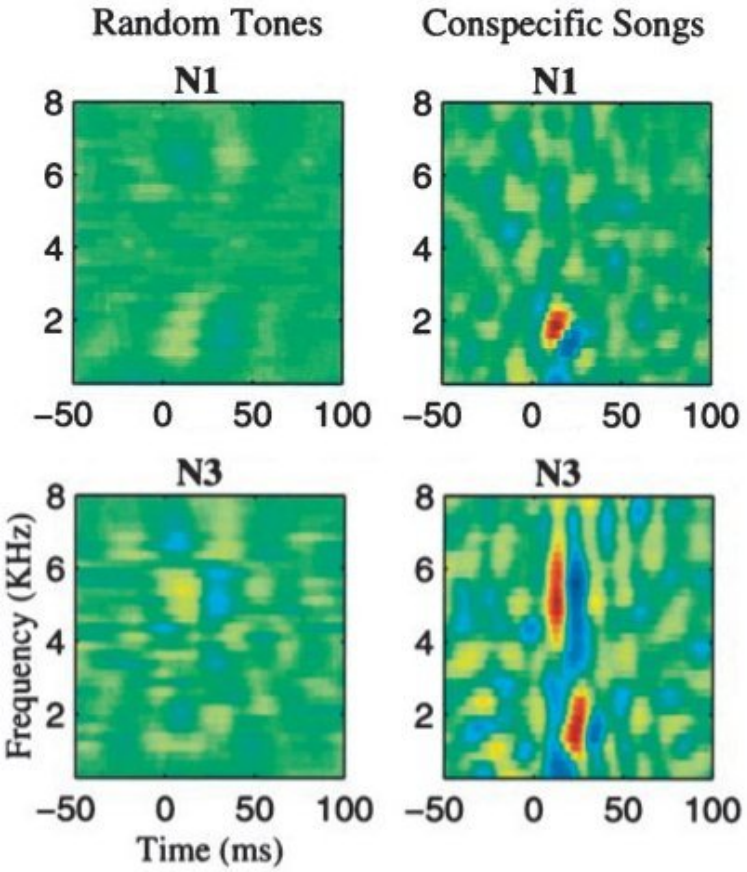


Figure 5.4: Spectro-temporal receptive fields of neurons in the avian auditory forebrain derived using random tone beeps (left) and a bird song ensemble (right). The STRFs differ both in amplitude and in shape. The top neuronal site N1 can be described as being sensitive to a moving spectral edge. The bottom neuronal site N3 is sensitive to a temporal combination of a high-frequency sound followed by a low-frequency sound. Both of these complex spectral-temporal responses are evident only when the song ensemble is used (from Theunissen et al., 2000).

pothesis in the auditory domain. If a model of the auditory cortex based on the slowness principle is able to reproduce some of the properties of the STRFs estimated from neurons probed with stimuli which obey the statistics of natural sounds, we may conclude that slowness is a well founded principle for the overall self-organization of the cortex as a whole.

5.2 Sound Representation

Physically, sound waves are longitudinal air compression waves $p(x,t)$, where air particles displacement $\psi(x,t)$ happens in the direction of wave motion:

$$\psi(x,t) = A\sin\left[\omega\left(\frac{x}{c} - t\right)\right] \tag{5.1}$$

$$p(x,t) = -\rho_0 c^2 \frac{\partial\psi}{\partial x} = -Z\omega A\cos\left[\omega\left(\frac{x}{c} - t\right)\right] \tag{5.2}$$

where ρ_0 is the undisturbed equilibrium density of the medium (air), Z the acoustic impedance of the medium, and c the speed of sound. Important physical quantities of a sound wave are the energy density

$$E = \rho_0\omega^2 A^2\cos^2\left[\omega\left(\frac{x}{c} - t\right)\right] \quad , \tag{5.3}$$

and the intensity, which is defined as the energy which flows per unit time across a unit area perpendicular to the wave propagation

$$I = cE = Z\omega^2 A^2\cos^2\left[\omega\left(\frac{x}{c} - t\right)\right] . \tag{5.4}$$

The standard threshold of hearing for humans is defined as the minimum intensity of a pure tone at 1000 Hz which a human can hear. It is given in terms of pressure as $P_0 = 20\mu$Pa or intensity as $I_0 = 10^{-12}\text{w/m}^2$ and it is a billionth of the atmospheric pressure. The threshold of hearing is usually used as a reference for values of intensity measured on a decibel scale:

$$I(\text{dB}) = 10\log_{10}\left[\frac{I}{I_0}\right] = 10\log_{10}\left[\frac{P^2}{P_0^2}\right] . \tag{5.5}$$

The dynamic range of the human auditory system can be defined as the span of intensities between the threshold of hearing and the threshold of pain, which is the intensity of a pure tone at 1000 Hz which causes pain when listened to. The threshold of pain is generally assumed to be 130 dB, which means that the dynamic range of human hearing spans 13 orders of magnitude, as opposed to the 7 to 10 order of magnitudes of the visual system (Ferwerda et al., 1996). Note that the intensity of the sound wave is *not* the perceived loudness. The relation between intensity (a physical quantity) and loudness (a psychological quantity) is given by the equal-loudness contours in Figure 5.5.

A straightforward visualization of a sound is a simple two dimensional plot where the time evolution of the air pressure wave is drawn (see Figure 5.6). The air pressure wave is often used to store digitalized audio signals. A typical audio signal at CD-quality is a

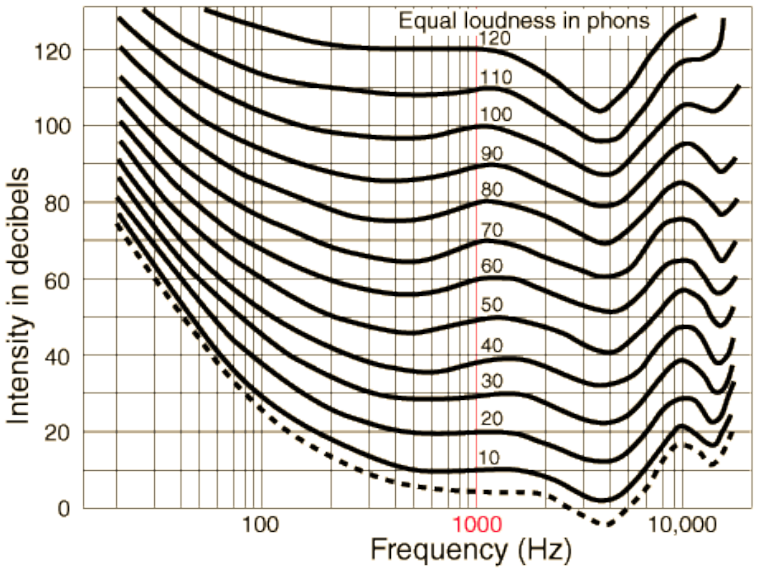


Figure 5.5: Equal-loudness contours (from `http://is.gd/g9WTW`).

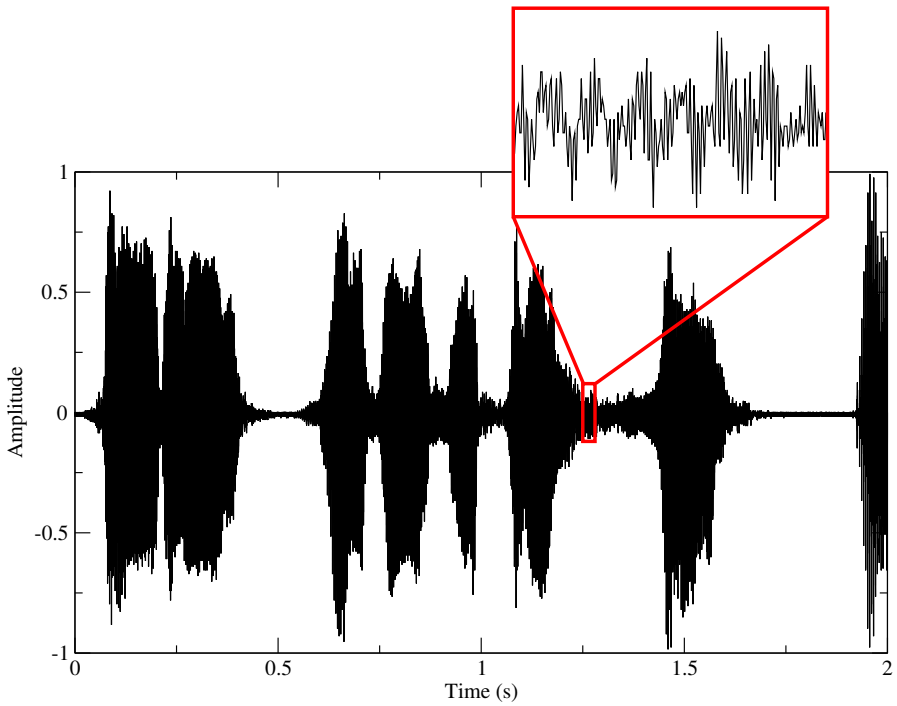


Figure 5.6: Representation of a sound wave as amplitude vs. time. Note the fine structure of the signal in the insert.

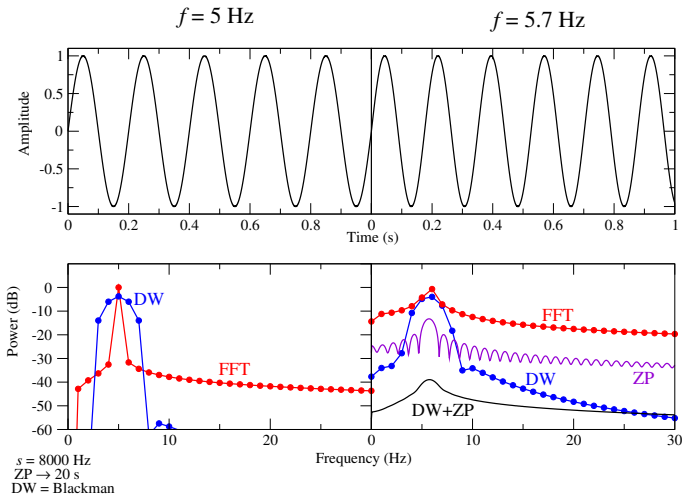


Figure 5.7: The leakage of frequency energy in Fourier space. In the top panel two sine waves with slightly different frequencies are show. In the bottom panel the FFT of these twon sines is shown. The raw FFT obtained with a sampling rate of 8000 Hz is shown in red, the FFT obtained applying a Blackman window is shown in blue, and the FFT obtained after zero-padding the signal is shown in violet.

stereo signal sampled at 44100 Hz, where the air pressure wave is encoded linearly in a 16 bit field.

The intensity of sound signals is often represented in the frequency domain by using the Fourier transform. Because signals are often digitalized, i.e. discrete samples are taken off the analog signal at a certain sampling rate s, the frequency representation of a sound signal obeys a set of constraints. The so-called *Nyquist critical frequency* is the maximum frequency that can be represented in Fourier space and is defined by $f_c = \frac{s}{2}$. The *frequency resolution*, i.e. the minimum frequency difference that can be represented is given by $\frac{s}{N} = \frac{1}{T}$, where N is the number of samples and T is the time length of the signal. Those constraints are nicely summarized in the concept of *acoustical quantum*, which results from the uncertainty relation first formulated by Gabor (1947):

$$\Delta t \cdot \Delta f \geq 1, \tag{5.6}$$

where in our context Δt is the sampling rate and Δf is the frequency resolution.

Frequencies which are present in the analog signal but can not be represented in the digital signal give raise to the *spectral leakage* effect, i.e. the spreading of the energy of non-representable frequencies to the neighboring representable frequencies, and to the *frequency aliasing* effect, i.e. the energy of frequencies higher than f_c gets assigned to very low frequencies. To cope with these effects different techniques are available, among

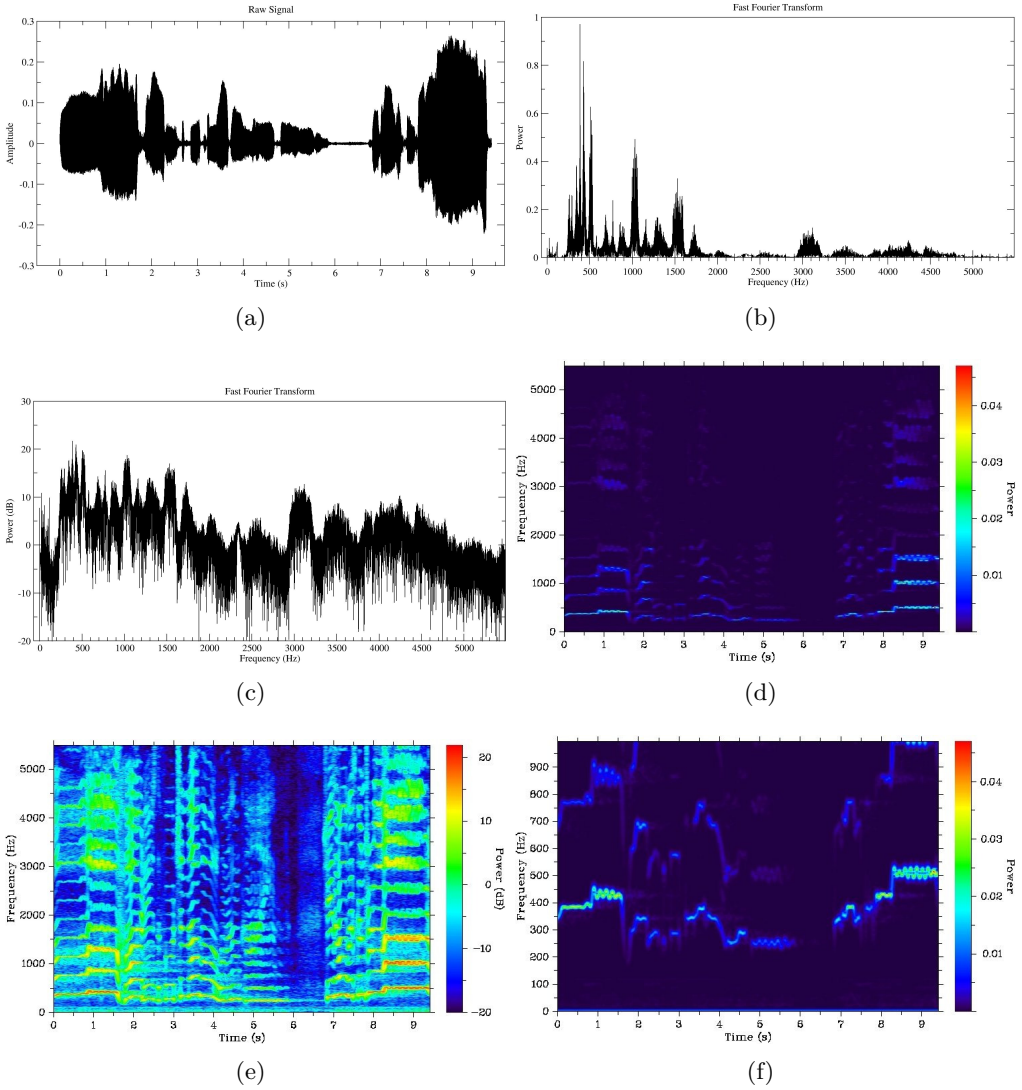


Figure 5.8: Sound representations: a) raw signal; b) FFT, linear scale; c) FFT, logarithmic scale; d) spectrogram [chunk size 8192, overlap 8000, linear]; e) spectrogram [chunk size 8192, overlap 8000, logarithmic]; f) spectrogram [chunk size 8192, overlap 8000, linear, lower frequency range].

which the so-called *zero-padding*, which consists in add a series of zero samples to the end of the digitized signal (to artificially "enhance" the frequency resolution), and the *data windowing*, which consists in the overlap of a smoothing window on top of the original signal to smooth the frequency leakage. An example is shown in Figure 5.7 A

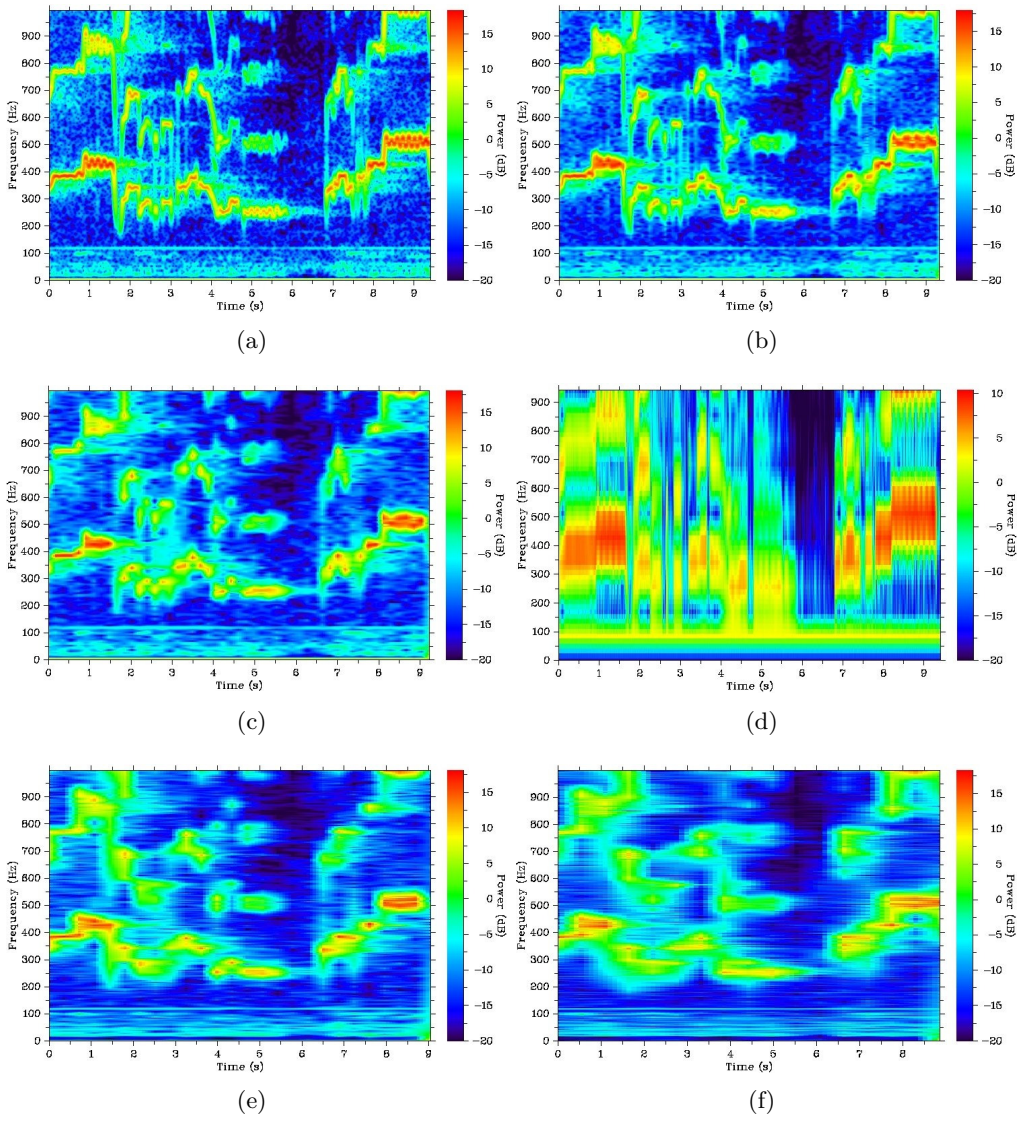


Figure 5.9: Sound representations: a) spectrogram [chunk size 8192, overlap 8000, logarithmic, lower frequency range]; b) spectrogram [chunk size 8192, overlap 4096, logarithmic, lower frequency range]; c) spectrogram [chunk size 8192, overlap 1, logarithmic, lower frequency range]; d) spectrogram [chunk size 512, overlap 256, logarithmic, lower frequency range]; e) spectrogram [chunk size 16384, overlap 1, logarithmic, lower frequency range]; f) spectrogram [chunk size 32768, overlap 8192, logarithmic, lower frequency range];

representation which resembles that used in the auditory system can be obtained with a mixed time-frequency representation, the short-time Fourier transform (STFT). Signal samples are grouped into overlapping chunks on which a Fourier transform is applied. The elements of the resulting complex matrix are multiplied with their complex conjugate and the logarithm of the resulting amplitude matrix is visualized as an image, the so-called *spectrogram*. The overlap between adjacent chunks is used to reduce the artifacts induced by the finite size of the chunks. In Figure 5.8 and Figure 5.9 spectrograms generated from the same source with different parameters and plotting scales are shown. These figures are meant to visualize the trade-off between the time and the frequency precision induced by the above mentioned uncertainty relation.

5.3 The *sonogram*

Even if the STFT representation is more similar to the kind of processing happening in the first stages of the human auditory system, more realistic models of those stages are available. For example, the STFT completely disregards phase information, and reconstruction of the original sound is not strightforward without this information and without making further assumptions (see for example Achan, Roweis, and Frey, 2003). In other words, there are infinitely many sound signals that have the same spectrogram. A successful computational model of the early stages of auditory processing based on neurophysiological, biophysical, and psychoacoustical results has been described by Shamma and co-workers in several papers (Yang, Wang, and Shamma, 1992; Elhilali, Chi, and Shamma, 2003). The model consists of the three stages depicted in Figure 5.10:

- The basilar membrane is modeled as a bank of asymmetric bandpass filters equally spaced on a logarithmic frequency axis. A total of 128 filters spanning a 5.2 octave range (0.1–4 kHz) are employed.

- To mimic the transduction step in the hair cells each filter output is half-wave rectified and low-pass filtered

- The mid-brain processing relays are modeled by a first-difference operation across the channel array, followed by a short-term integration. The result is a sharpening of the bandwidths of the cochlear filter

The result of the above sequence is the *auditory spectrogram* or *sonogram*: a spectro temporal representation of a sound signal that resembles a plain STFT, but in addition to mimicking the early stages of the auditory pathway, it allows for a (lossy) reconstruction of the original signal by means of convex projections (Youla and Webb, 1982; Mallat, 1989).

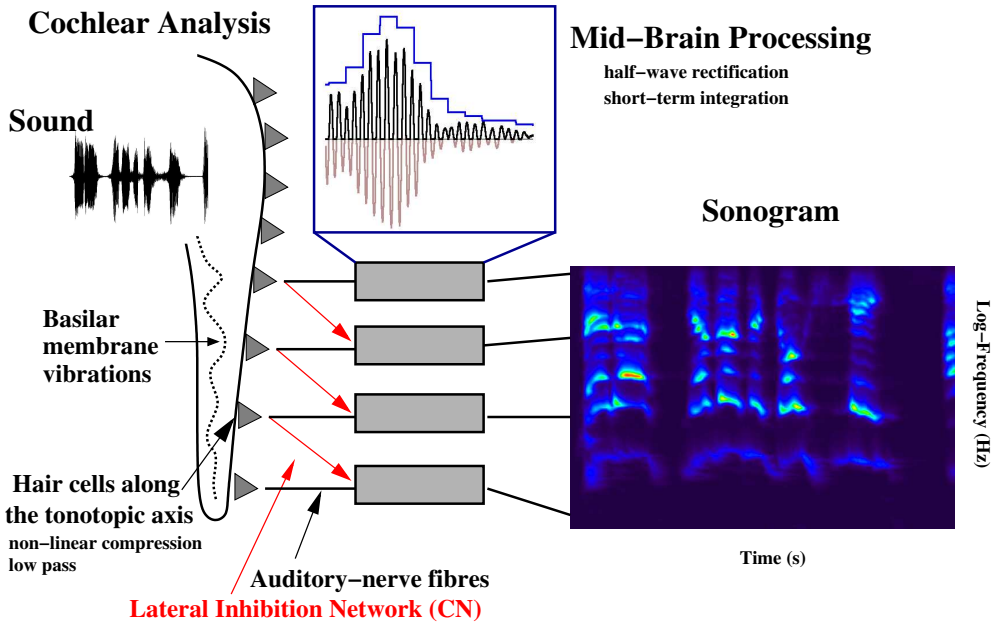


Figure 5.10: The Shamma model of the early stages of auditory processing (from Elhilali et al., 2003).

5.4 A Computational Model of the Auditory Cortex

5.4.1 SFA on the sound wave

A model of the auditory cortex based on the slowness principle inspired by the successful results in the visual cortex must take into account the fundamental difference of the input stimulus in the auditory and visual modalities. Sound is inherently one dimensional, but the implementation of the slowness principle, namely the Slow Feature Analysis (SFA) algorithm (see Section 2.2), assumes a multidimensional signal. In Wiskott (2003) *time-embedding* is used to overcome this problem: A window of N adjacent one-dimensional input samples $\mathbf{s}_1 = (s(1), s(2), \ldots, s(N))^T$ is considered as a N-dimensional input vector; the next input vector results from the forward sliding of the window $\mathbf{s}_2 = (s(1+g), s(2+g), \ldots, s(N+g))^T$, where g is the gap between two adjacent windows, which defines the *speed* of the sliding window. The problem of this approach in the processing of audio signals is that the huge number of samples in a window that are needed to collect information at a time scale that is relevant for the auditory system makes it computationally impractical, if not impossible.

We performed some preliminary experiments by applying SFA directly to the audio signal in the air pressure wave representation. Input signal is a synthetic song, where random harmonic tones are played. Every tone was enveloped in a window to allow for

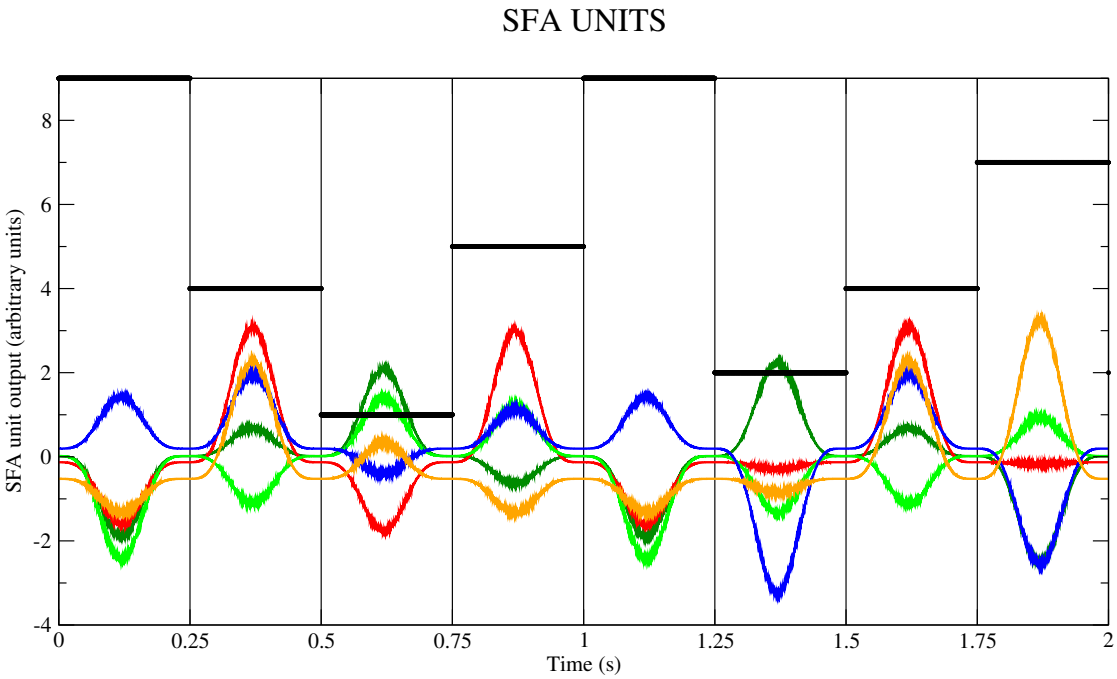


Figure 5.11: The output of some of the SFA units trained on a sequence of tones is shown. Different colors correspond to different units. The black straight lines show the tones that were played as input signal.

smooth transitions. The input signal then gets embedded in time as described above, filtered through a PCA step to reduce dimensionality, expanded in a quadratic polynomial space, and finally SFA units are learned in the high dimensional space. The procedure has been described already in Sections 2.2 and 2.2.1. The results are shown in Figure 5.11. The scatter plot of unit pairs in Figure 5.12 helps in visualizing the relevance of the result: The simple linear patterns emerging mean that two SFA units code for all the 10 tones presented. This is shown in more detail in the bottom panel in Figure 5.12: most of the errors in tone detection are concentrated at transitions, where the smoothing window forces the signal to be near zero. i.e. not detectable. This approach still works if the input signal is a frequency modulated pure tone: In the top panels of Figure 5.13 the STFT of such a signal and the output of the first SFA unit is shown.

However, this approach shows its limits as soon as more complicated input signals are presented. In the bottom panels of Figure 5.13 the STFT of a frequency modulated pure tone signal is shown together with the output of the first SFA units. In this case the modulation is a random walk. The SFA units are clearly not picking up the random walk.

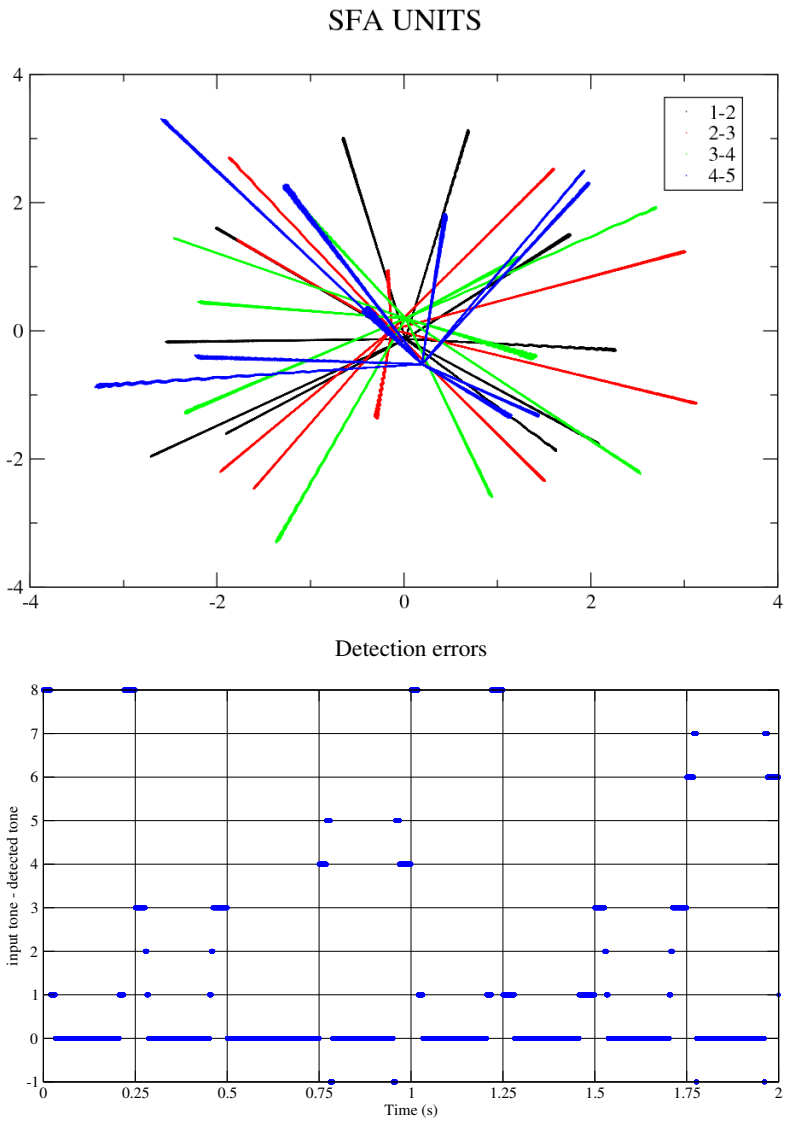


Figure 5.12: The output of some SFA unit pairs is displayed in a scatter plot (top). The tone detection error is displayed in the bottom panel.

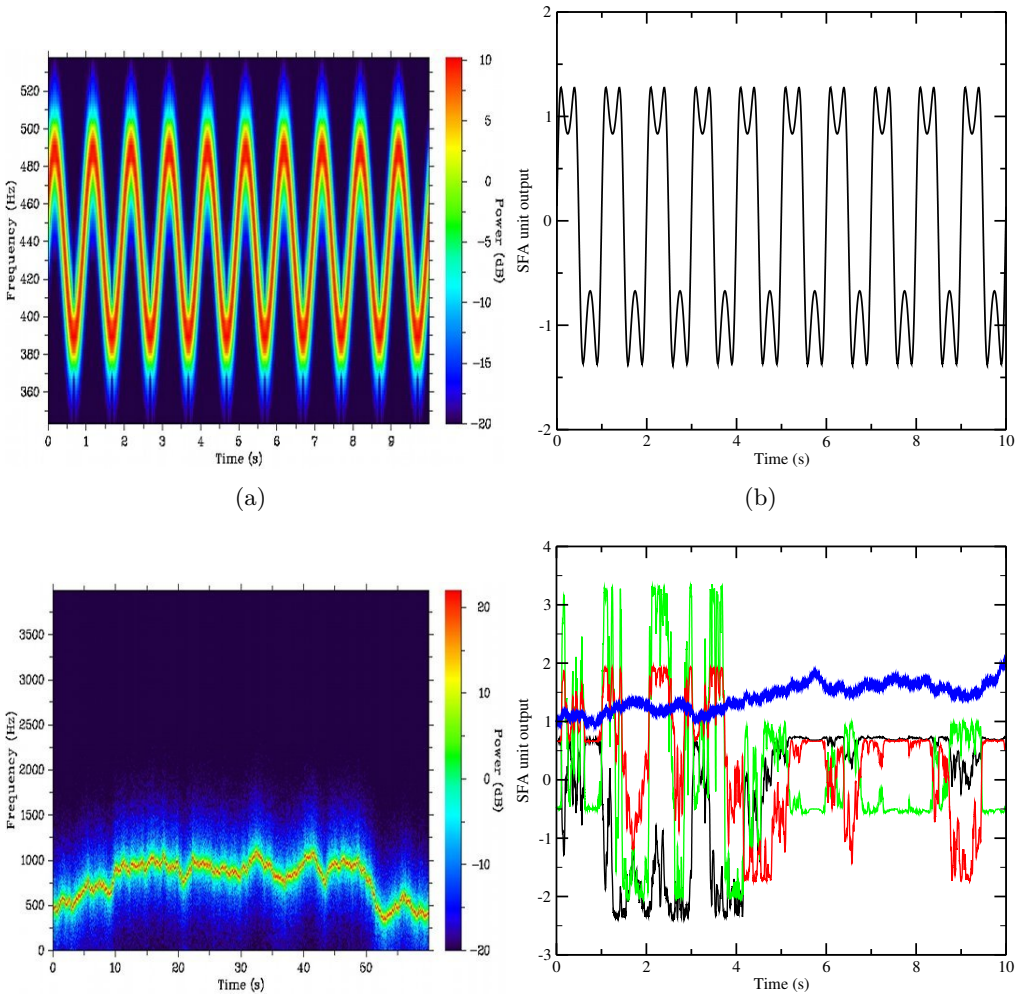


Figure 5.13: Spectrogram of a frequency modulated pure tone (top-left) and output of the slowest SFA unit (top-right): the units somehow picks up the modulation signal, but with a strange inversion of the tips in the output signal. If the modulation is a random walk (bottom-left) none of the units (bottom-right) seem to relate to the frequency modulation (shown in blue).

5.4.2 SFA on the *sonogram*

More tests have been performed, with multiple amplitude and frequency modulations, using more realistic sounds, including music and speech, but the main result remains: The sound wave representation, although simple and straightforward, is not usable as an input to the SFA algorithm. A more preprocessed representation may be more suited. Another reason to use a different representation is that it is more biologically plausible: The auditory nerve neurons already receive a signal that is decomposed in the frequency and time domain in analogy with a STFT. The transformation of the signal from a one-dimensional wave to a multidimensional frequency/time signal is performed by the interplay of mechanical and hydrodynamical forces within the cochlear duct. We choose the model of the early stages of the auditory processing by Shamma and coworkers presented in Section 5.2 as a preprocessing step. The model has been implemented from scratch in Python using the Modular toolkit for Data Processing presented in Chapter 6.

The model of the auditory cortex based on the slowness principle we implemented thus comprises four steps:

1. The input signal is preprocessed through Shamma's filter bank to generate an auditory spectrogram (Figure 5.10).

2. A principal component analysis is performed on the auditory spectrogram along the frequency axis, to reduce the number of spectral dimensions (Figure 5.14).

3. An $M \times N$ window is slid on the dimensionality reduced auditory spectrogram. This window is slided along the time axis. The result is a sequence of $M \times N$ pixel matrices. Those matrices are then cast to $(M \cdot N)$-dimensional vectors. A principal component analysis is then performed on the space of the resulting vectors. This step is used to reduce dimensionality both in the spectral as well as in the time dimension (Figure 5.15).

4. The output of the principal components is then expanded in a quadratic polynomial space and slow feature analysis is performed (Figure 5.16).

As an input signal the audio-book "Das Parfum" by Patrick Süskind read by Gert Westphal (ISBN-13: 978-3894699116) has been used. The stereo audio signal has been averaged into a mono channel, down-sampled to 8000 Hz without noticeable difference in the quality of the resulting sound. The total number of samples was exceeding 250 thousands for a total of more than 9 hours audio signal. The model requires the specification of several parameters, e.g. the size of the time window, the parameters for the Shamma pre-processing step (wavelet frame length, integration time, non-linearity factor), the number of principal components to keep after PCA, the number of SFA units to train. Several parameter combinations have been used, but the main results are qualitatively comparable. In particular, the principal components shown in Figure 5.15

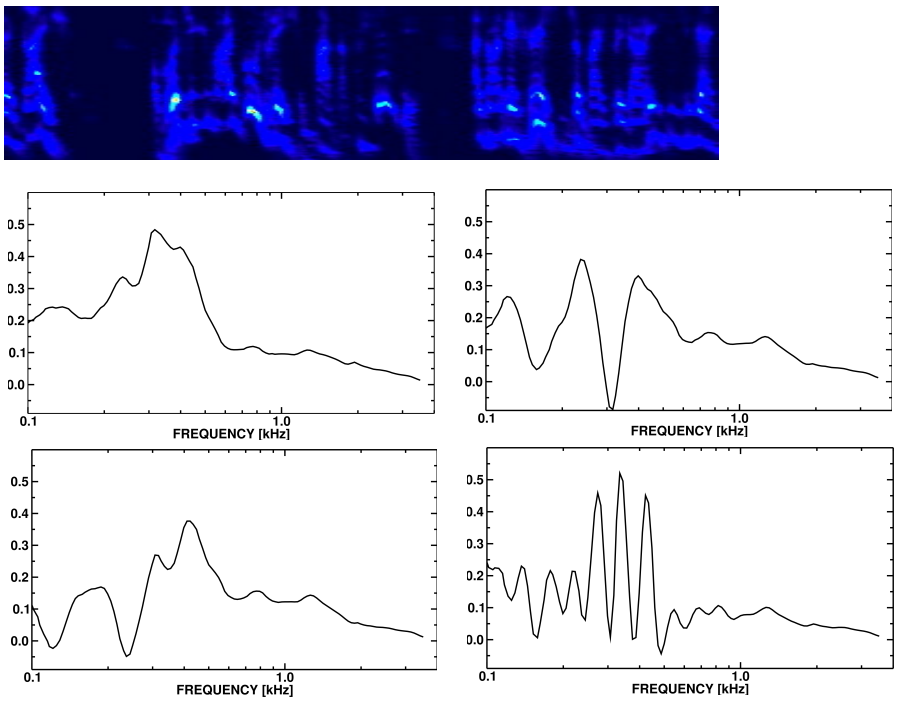


Figure 5.14: Model of the auditory cortex. Step 2. The auditory spectrogram is filtered through the principal components trained on the spectral axis. The first four principal components are shown.

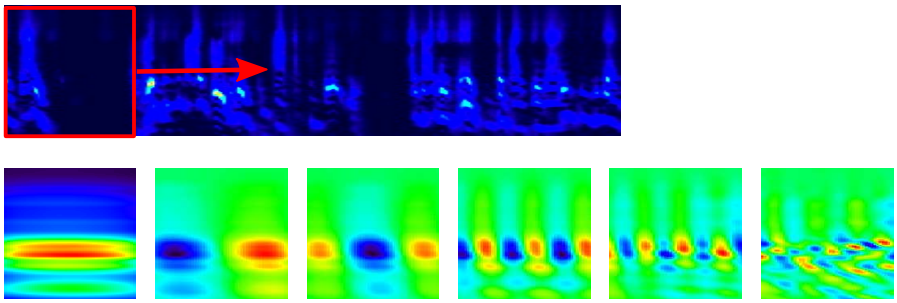


Figure 5.15: Model of the auditory cortex. Step 3. The auditory spectrogram shown at the top is a reconstruction in the input space of the output of step 3. This gets windowed and time-embedded. Principal component analysis is performed on the resulting vector space. The first seven principal components are shown (back projected in input space for clarity).

are quite stable and they have a very regular structure: They represent sounds that vary slowly in both the spectral and time direction. Very similar results have been obtained

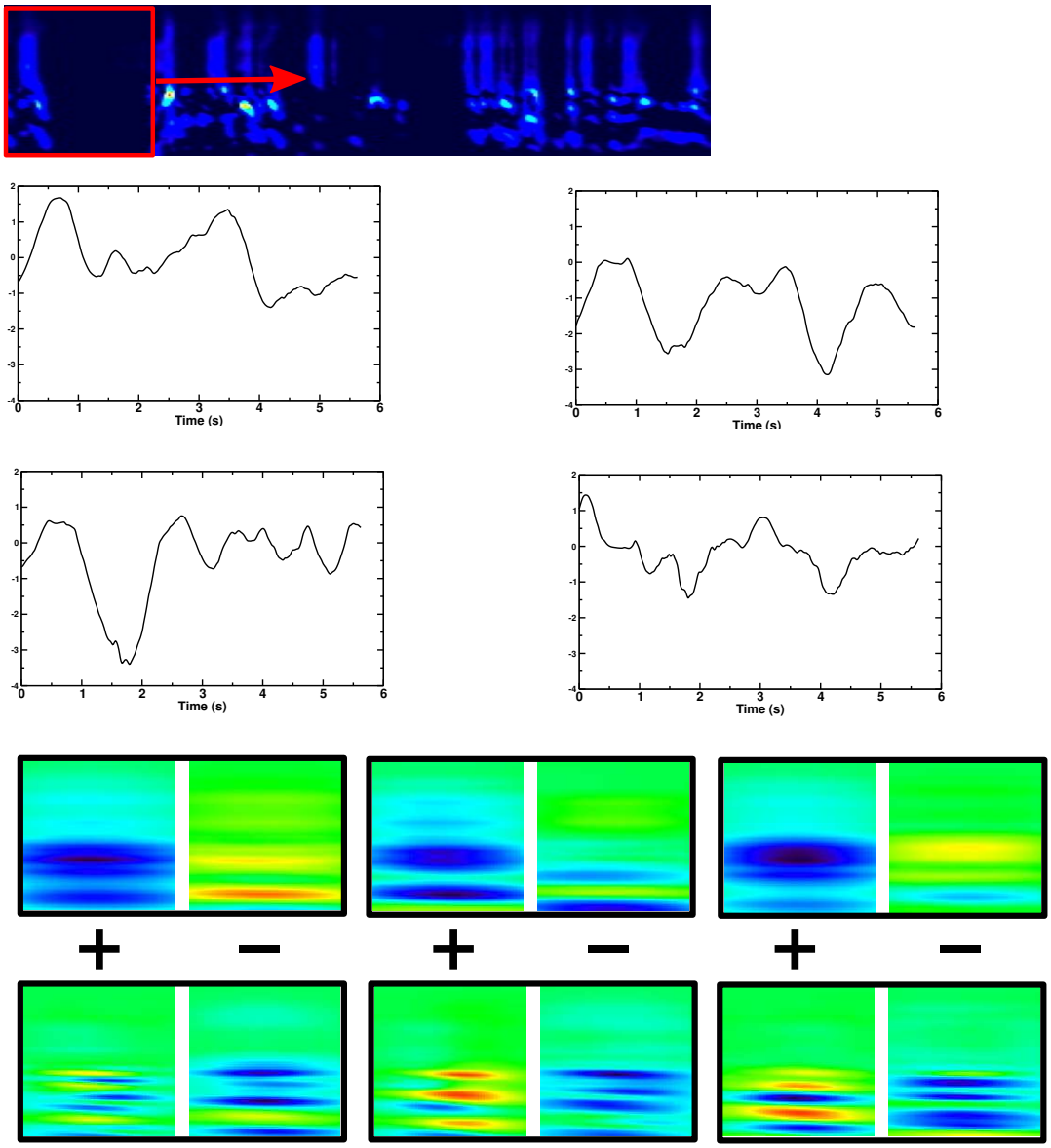


Figure 5.16: Model of the auditory cortex. Step 4. The auditory spectrogram shown on top is a reconstruction in the input space of the output of step 3. Slow feature analysis is performed. The output of the first 4 slow features is shown in the middle panel. In the bottom panel optimal excitatory (denoted by a "+" sign) and inhibitory stimuli (denoted by a "−" sign) pairs are shown for the first six slow features (back projected in input space for clarity).

by Körding, König, and Klein (2002) in a model for learning sparse receptive fields in the auditory domain.

The problem of the interpretation of the results for the SFA units has been discussed in Berkes and Wiskott (2003). The same algorithm has been used here. A good way to characterize those units is to find out the optimal excitatory and inhibitory stimuli, i.e. the input that triggers the highest responses (in absolute value). These inputs should then resemble the receptive fields, in our case the spectro-temporal receptive fields, of the modeled neurons. A comparison of the optimal stimuli of the trained SFA units in Figure 4 with the measured STRF in Figure 5.4 show that the result is negative. The SFA units seem to prefer signals with constant spectral content and almost constant in time, whereas the STRF are typically well localized in time and frequency. This is unfortunate but most probably inevitable if slowness is used as the only learning principle for the auditory cortex: Slowness always favors signal components that are varying slowly, but it is probably in the transients and in the fine-structure in the spectro-temporal domain that most of the behaviorally relevant information is retained. A combination of slowness with sparseness may yield better results. No further attempts have been made to adapt the model to cope with such difficulties. The question if slowness is a valid computational principle for the organization of the cortex beyond the visual domain remains open.

6 The Modular toolkit for Data Processing

6.1 Python in neuroscience

The use of the Python programming language in computational neuroscience has been growing steadily during the past few years. The maturation of two important open source projects, the scientific libraries NumPy[1] and SciPy[2], gives access to a large collection of scientific functions that rivals in size and speed well known commercial alternatives like The MathWorks™ Matlab®[3]. Furthermore, the flexible and dynamic nature of Python offers the scientific programmer the opportunity to quickly develop efficient and structured software while maximizing prototyping and reusability capabilities.

In this chapter we describe the Modular tollkit for Data Processing (MDP), which I developed together with Dr. Pietro Berkes and Niko Wilbert as an essential tool for all the computations presented in this thesis. This work has been published in Zito, Wilbert, Wiskott, and Berkes (2008).

6.2 Introduction to MDP

The Modular toolkit for Data Processing (MDP) package[4] contributes to the growing community of Python in neuroscience a library of widely used data processing algorithms and the possibility to combine them according to a pipeline analogy to build more complex data processing software.

MDP has been designed to be used as-is and as a framework for scientific data processing development. From the user's perspective, MDP consists of a collection of supervised and unsupervised learning algorithms and other data processing units (*nodes*) that can be combined into data processing sequences (*flows*) and more complex feed-forward network architectures. Given a set of input data, MDP takes care of successively training or executing all nodes in the network. This allows the user to specify complex algorithms as a series of simpler data processing steps in a natural way. The base of available algorithms is steadily increasing and includes, to name but the most common, Principal Component Analysis (PCA and NIPALS), several Independent Component Analysis

[1] `http://numpy.scipy.org`
[2] `http://www.scipy.org`
[3] `http://www.mathworks.com/products/matlab/`
[4] `http://mdp-toolkit.sourceforge.net`

algorithms (CuBICA, FastICA, TDSEP, and JADE), Locally Linear Embedding, Slow Feature Analysis, Gaussian Classifiers, Fisher Discriminant Analysis, Factor Analysis, and Restricted Boltzmann Machine (see Table 6.1 for a more exhaustive list and references). Particular care has been taken to make computations efficient in terms of speed and memory. To reduce memory requirements, it is possible to perform learning using batches of data and to define the internal parameters of the nodes to be single precision, which makes the usage of very large data sets possible. Moreover, an MDP subpackage in its final stages of development offers a parallel implementation of the basic nodes and flows.

From the developer's perspective, MDP is a framework that makes the implementation of new supervised and unsupervised learning algorithms easy and straightforward. The basic class, `Node`, takes care of tedious tasks like numerical type and dimensionality checking, leaving the developer free to concentrate on the implementation of the learning and execution phases. Because of the common interface, the node then automatically integrates with the rest of the library and can be used in a network together with other nodes. A node can have multiple training phases and even an undetermined number of phases. This allows the implementation of algorithms that need to collect some statistics on the whole input before proceeding with the actual training, and others that need to iterate over a training phase until a convergence criterion is satisfied.

MDP is distributed under the open source LGPL license. It has been written in the context of theoretical research in neuroscience, but was designed to be helpful in any context where trainable data processing algorithms are used. Its simplicity on the user's side together with the reusability of the implemented nodes make it also a useful educational tool.

6.3 The package structure

The MDP framework consists of a library of data processing nodes with a common Application Programming Interface (API) and a collection of objects which are used to connect nodes together to implement complex data processing workflows. In the following sections the framework structure is outlined followed by an example application. The full API together with an extensive tutorial covering both usage and instruction for writing extensions are available from the MDP homepage.

6.3.1 Nodes

A *node* is the basic building block of an MDP application. It represents a data processing element, like for example a learning algorithm, a data filter, or a visualization step (see Table 6.1 for a list of some of the available algorithms). Each node is characterized by an input dimension (i.e., the dimensionality of the input vectors), an output dimension,

Node class name	Algorithm & Reference
`PCANode`	Principal Component Analysis Jolliffe (1986)
`NIPALSNode`	Nonlinear Iterative Partial Least Squares PCA (NIPALS) Fritzke (1995)
`CuBICANode`	Cumulant-based Independent Component Analysis (CuBICA) Blaschke and Wiskott (2004)
`FastICANode`	Independent Component Analysis (FastICA) Hyvärinen (1999)
`JADENode`	Cumulant-based Independent Component Analysis (JADE) Cardoso (1999)
`TDSEPNode`	Temporal blind-source separation algorithm (TDSEP) Ziehe and Müller (1998)
`LLENode`	Locally Linear Embedding Analysis Roweis and Saul (2000)
`HLLENode`	Hessian Locally Linear Embedding Analysis Donoho and Grimes (2003)
`FDANode`	Fisher Discriminant Analysis Bishop (1995)
`SFANode`	Slow Feature Analysis Wiskott and Sejnowski (2002)
`ISFANode`	Independent Slow Feature Analysis Blaschke et al. (2007)
`RBMNode`	Restricted Boltzmann Machine Hinton et al. (2006)
`GrowingNeuralGasNode`	Growing Neural Gas (learn a graph structure of the data) Fritzke (1995)
`FANode`	Factor Analysis Bishop (2007)
`GaussianClassifierNode`	Supervised gaussian classifier
`PolynomialExpansionNode`	Expand the signal in a polynomial space
`TimeFramesNode`	Expand the signal using a sliding temporal window (temporal embedding)
`HitParadeNode`	Record local minima and maxima in the signal
`NoiseNode`	Additive and multiplicative noise injection

Table 6.1: Some of the nodes available in MDP.

and a `dtype`, which determines the numerical type of the internal structures and of the output signal. By default, these attributes are inherited from the input data.

Nodes can have a *training phase*, where training data is analyzed in order to adapt the internal variables, and an *execution phase*, where new data can be processed using the learned parameters. For example, the Principal Component Analysis (PCA) algorithm (Jolliffe, 1986) requires the computation of the mean and covariance matrix of a set of training data from which the principal eigenvectors of the data distribution are estimated. MDP offers an implementation of this algorithm in the class `PCANode`. The node

can be trained on the data using the interface common to all nodes: `PCANode.train(x)` analyzes a new batch of data `x`, and updates the estimation of mean and covariance matrix; `PCANode.stop_training()` finalizes the algorithm by computing and selecting the principal eigenvectors. Once the training is finished, new data can be projected on the principal components calling the `PCANode.execute(y)` method. If the transformation specified by the underlying algorithm is invertible, the node can also be executed "backwards" using the `PCANode.inverse(z)` method. In the case of PCA, for example, this corresponds to projecting a vector in the principal components space back to the original data space.

`Node` was designed to be applied to arbitrarily long sets of data: if the underlying algorithms support it, the internal structures can be updated incrementally by sending multiple batches of data. It is thus possible to perform computations on amounts of data that would not fit into memory or to generate data on-the-fly. The general form of the training phase thus is:

```
# create an instance of the desired node
node_instance = mdp.nodes.XXXNode()

for data_batch in data_source:
   node_instance.train(data_batch)

node_instance.stop_training()
```

In the code, `data_source` can be any Python iterator[5] (e.g., a list, an iterator object, or a generator function) that returns an array with a batch of training data. The last line finalizes the training phase. It is shown here for completeness, but can be replaced by a call to the `execute` or `inverse` methods. Nodes also define some utility methods, like for example `copy` and `save`, that return an exact copy of a node and save it in a file, respectively. Additional methods may be present, depending on the algorithm. The `PCANode.get_projmatrix` method, for example, returns the matrix projecting input data into the principal components' space. For a toy signal-denoising application that makes use of the basic `Node` features just described see Figure 6.1.

Some nodes, namely the one corresponding to supervised algorithms, e.g. Fisher Discriminant Analysis (Bishop, 1995), may need some labels or other supervised signals to be passed during training:

```
input = {'a': data_a, 'b':data_b, 'c':data_c}
fdanode = mdp.nodes.FDANode()
for label in ['a', 'b', 'c']:
    fdanode.train(input[label], label)
```

[5] `http://docs.python.org/lib/typeiter.html`

```
# Simple denoising algorithm
# Given is a set of multidimensional signals, for example
# EEG waves, from which normal statistics are learned,
# and a set of noisy signals to be denoised.

# 1 - Create an instance of the PCA algorithm
#     The argument output_dim = 0.9 tells the node to retain
#     a number of principal components such that the
#     explained variance is at least 90%.
#     A fixed number of output components can be specified
#     for example by output_dim=10
pcanode = mdp.nodes.PCANode(output_dim = 0.9)

# 2 - Perform PCA on the set of training signals
pcanode.train(signals)

# 3 - Stop learning and estimate the principal components
pcanode.stop_training()

# 4 - Project noisy signals in to the principal component space
proj_signals = pcanode.execute(noisy_signals)

# 5 - Project the data back to the input space for visualization
#     and comparison with original data
denoised_signals = pcanode.inverse(proj_signals)
```

Figure 6.1: A simple denoising application.

A node could also require multiple training phases. For example, the training of `fdanode` is not complete yet, since it has two training phases: The first one computing the mean of the data conditioned on the labels, and the second one computing the overall and within-class covariance matrices and solving the FDA problem. The first phase must be stopped and the second one trained:

```
fdanode.stop_training()
for label in ['a', 'b', 'c']:
    fdanode.train(input[label], label)
```

The easiest way to train multiple phase nodes is using flows, which automatically handle multiple phases (see Section 6.3.2).

MDP makes it easy to write new nodes that interface with the existing data processing elements. The `Node` class is designed to make the implementation of new algorithms easy and intuitive. This base class takes care of setting input and output dimension and casting the data to match the numerical type (e.g. float or double) of the internal

```
class MeanFreeNode(mdp.Node):
     def __init__(self, input_dim=None, dtype=None):
         super(MeanFreeNode, self).__init__(input_dim=input_dim,
                                            dtype=dtype)
         self.avg = None
         self.tlen = 0

     def _train(self, x):
         # Initialize the mean vector with the right
         # size and dtype if necessary:
         if self.avg is None:
             self.avg = mdp.numx.zeros(self.input_dim,
                                       dtype=self.dtype)
         # Update the average
         self.avg += mdp.numx.sum(x, axis=0)
         # Update the number of data points examined
         self.tlen += x.shape[0]

     def _stop_training(self):
         # Compute the average signal
         self.avg /= self.tlen

     def _execute(self, x):
         return x - self.avg

     def _inverse(self, y):
         return y + self.avg
```

Figure 6.2: Definition of a new node that removes the mean of the signal.

variables, and offers utility methods that can be used by the developer. To expand the MDP library of implemented nodes with user-made nodes it is sufficient to subclass `Node`, overriding some of the methods according to the algorithm one wants to implement, typically the `_train`, `_stop_training`, and `_execute` methods. Figure 6.2 shows an example of a simple node that removes the mean of the signal. A more detailed introduction to writing new nodes in MDP can be found in the online tutorial[6].

It is also possible to specify multiple training phases by defining additional training methods and overwriting the `_get_train_seq` method. For example

```
class MultiplePhaseNode(mdp.Node):
   def _get_train_seq(self):
      return [(self._train_A, self._stop_A),
              (self._train_B, self._stop_B)]
```

[6] `http://mdp-toolkit.sourceforge.net/tutorial.html`

defines a new node with two training phases, one updated by the method `_train_A` and finalized using `_stop_A`, and analogously the second is defined by the methods `_train_B` and `_stop_B`. The final user will still perform the training phase by calling the usual methods `train` and `stop_training` (although multiple times), and need not know about the specific implementation of the algorithm.

6.3.2 Flows

A *flow* is a sequence of nodes that are trained and executed together to form a more complex algorithm. Input data is sent to the first node and is successively processed by the subsequent nodes along the sequence. Using a flow as opposed to handling manually a set of nodes has a clear advantage: The general flow implementation automates the training (including supervised training and multiple training phases), execution, and inverse execution (if defined) of the whole sequence. For example, suppose we need to analyze a very high-dimensional input signal using Independent Component Analysis (ICA). To reduce the computational load, we would like to reduce the input dimensionality of the data using PCA. Moreover, we would like to find the data that produces local maxima in the output of the ICA components on a new test set (this information could be used for instance to characterize the ICA filters). To implement this algorithm using MDP, we need to generate an instance of `Flow` using the appropriate nodes:

```
# Define a data processing sequence.
# - PCANode(output_dim=5) performs PCA and keeps the
#                         first 5 principal components only
# - CuBICANode() is a cumulant-based ICA algorithm
# - HitParadeNode(3) records the 3 largest local maxima from the
#                    output of the previous node
flow = mdp.Flow([mdp.nodes.PCANode(output_dim=5),
                 mdp.nodes.CuBICANode(),
                 mdp.nodes.HitParadeNode(3)])
```

The training and execution are performed as for the `Node` class:

```
# Train all the nodes using the data array 'x'
flow.train(x)
# Compute the output of the node sequence when presented
# with array 'x_test'
output = flow.execute(x_test)
```

A single call to the flow's `train` method will automatically take care of training nodes with multiple training phases, if such nodes are present.

`Flow` objects are defined as Python containers, and thus are endowed with most of the methods of Python lists: One can obtain slices, append new nodes, pop or in-

sert nodes, and concatenate flows. For example, to get the maxima computed by the `HitParadeNode`, one can refer to the last node using the list construct `flow[-1]`:

```
maxima, indices = flow[-1].get_maxima()
```

The `Flow` class defines a number of utility methods, including `save` and `copy` methods. It also implements a crash recovery mechanism that can be activated by setting a flag: In case an exception is thrown during training, the current state of the flow is saved for later inspection.

6.3.3 Hierarchical networks

In case the desired data processing application cannot be defined as a sequence of nodes, the `hinet` subpackage makes it possible to construct arbitrary feed-forward architectures, and in particular hierarchical networks. It contains three basic building blocks (which are all nodes themselves): `Layer`, `FlowNode`, and `Switchboard`.

The first building block, `Layer`, works like a horizontal version of flow. It acts as a wrapper for a set of nodes that are trained and executed in parallel. For example, we can combine two nodes with 100 dimensional input to construct a layer with a 200-dimensional input:

```
node1 = mdp.nodes.PCANode(input_dim=100, output_dim=10)
node2 = mdp.nodes.SFANode(input_dim=100, output_dim=20)
layer = mdp.hinet.Layer([node1, node2])
```

The first half of the 200 dimensional input data is then automatically assigned to `node1` and the second half to `node2`. We can train and execute a layer just like any other node.

In order to be able to build arbitrary feed-forward node structures, `hinet` provides a wrapper class for flows (i.e., vertical stacks of nodes) called `FlowNode`. For example, we can replace `node1` in the above example with a `FlowNode`:

```
node1_1 = mdp.nodes.PCANode(input_dim=100, output_dim=50)
node1_2 = mdp.nodes.SFANode(input_dim=50, output_dim=10)
node1_flow = mdp.Flow([node1_1, node1_2])
node1 = mdp.hinet.FlowNode(node1_flow)
node2 = mdp.nodes.SFANode(input_dim=100, output_dim=20)
layer = mdp.hinet.Layer([node1, node2])
```

`node1` has two training phases in this example, one for each internal node. Therefore `layer` now has two training phases as well and behaves like any other node with two training phases. By combining and nesting `FlowNode` and `Layer`, it is thus possible to build complex node structures.

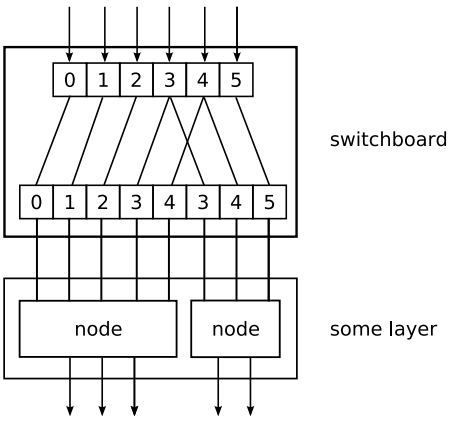


Figure 6.3: Example of feed-forward network topology.

When implementing networks one might have to route different parts of the data to different nodes in a layer in complex ways. This is done by the `Switchboard` node, which can handle such routing. A `Switchboard` is initialized with a 1-D array with one entry for each output connection, containing the corresponding index of the input connection that it receives its input from, e.g.:

```
switchboard = mdp.hinet.Switchboard(input_dim=6,
                                    connections=[0,1,2,3,4,3,4,5])
print switchboard
# should print: Switchboard(input_dim=6, output_dim=8, dtype=None)
x = mdp.numx.array([[2,4,6,8,10,12]])
print switchboard.execute(x)
# should print: array([[ 2,  4,  6,  8, 10,  8, 10, 12]])
```

The switchboard can then be followed by a layer that splits the routed input to the appropriate nodes, as illustrated in Figure 6.3.

Since hierarchical networks can become quite complicated to build and debug, `hinet` includes the class `HiNetHTML` that translates an MDP flow into a graphical visualization in an HTML file.

6.4 A complete application

In this section we show a complete example of MDP usage in a machine learning application and use non-linear Slow Feature Analysis for processing of non-stationary time series. We consider a chaotic time series derived by a logistic map (a demographic model of the population biomass of species in the presence of limiting factors such as food supply or disease) that is non-stationary in the sense that the underlying parameter

```
import mdp
N = mdp.numx

def logistic_map(x,r):
    return r*x*(1-x)

# time axis is 1 second sampled at 10KHz
t = N.linspace(0,1,10000,endpoint=0)

# driving force
dforce = N.sin(10*N.pi*t) + N.sin(22*N.pi*t) + N.sin(26*N.pi*t)

# resulting time series
series = N.zeros((10000,1),'d')
series[0] = 0.6 # initial condition
for i in range(1,10000):
    series[i] = logistic_map(series[i-1],3.6+0.13*dforce[i])

# define the flow
sequence = [mdp.nodes.EtaComputerNode(),
            mdp.nodes.TimeFramesNode(10),
            mdp.nodes.PolynomialExpansionNode(3),
            mdp.nodes.SFANode(output_dim=1),
            mdp.nodes.EtaComputerNode()]

flow = mdp.Flow(sequence, verbose=1)

# train the flow
flow.train(series)

# execute the flow to get the SFA estimate of the driving force
slow = flow.execute(series)

# rescale driving force to compare with SFA estimate
resc_dforce = (dforce - N.mean(dforce,0))/N.std(dforce,0)

# verify that the results are correct
# result should be > 0.99
print mdp.utils.cov2(resc_dforce[:-9],slow)
# result should be ~= 3000
print 'Eta value (time-series): ', flow[0].get_eta(t=10000)
# result should be ~= 10
print 'Eta value (slow feature): ', flow[-1].get_eta(t=9996)
```

Figure 6.4: Python code to reproduce the results in Wiskott (2003).

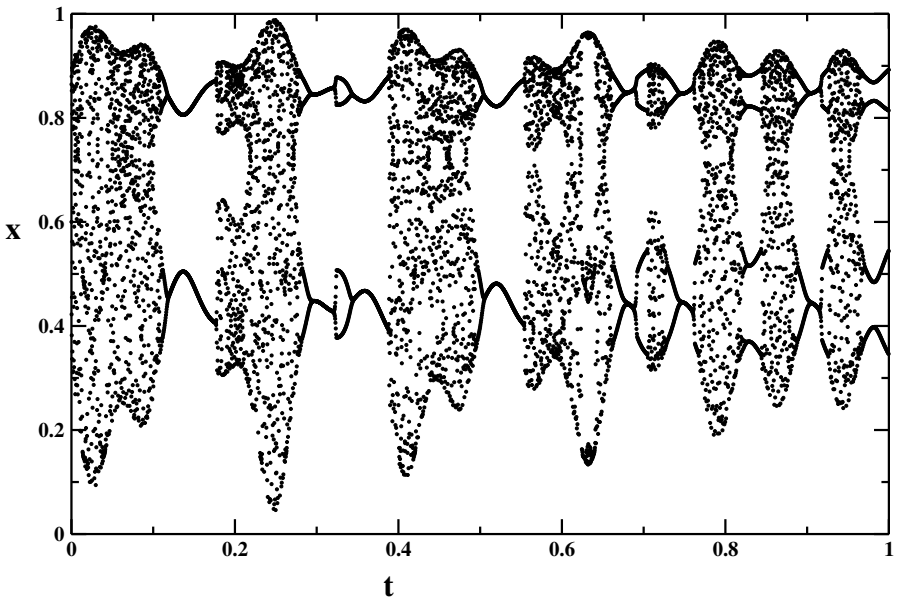


Figure 6.5: Chaotic time series generated by the logistic equation.

is not fixed but is varying smoothly in time. The goal is to extract the slowly varying parameter that is hidden in the observed time series. This example reproduces some of the results reported in Wiskott, 2003. The complete code is shown in Figure 6.4.

We first generate the slowly varying driving force parameter r_t as a combination of three sine waves $r_t = \sin(10\pi t) + \sin(22\pi t) + \sin(26\pi t)$. We then generate the time series using the logistic equation $x_{t+1} = (3.6 + 0.13r_t)x_t(1 - x_t)$. The resulting time series x is shown in Figure 6.5.

To reconstruct the underlying parameter, we define a `Flow` to perform SFA in the space of polynomials of degree 3. We first use a node that embeds the 1-dimensional time series in a 10 dimensional space using a sliding temporal window of size 10 (`TimeFramesNode`). Second, we expand the signal in the space of polynomials of degree 3 using the corresponding node. Finally, we perform SFA on the expanded signal and keep the slowest feature using the `SFANode`. In order to measure the slowness of the input time series before and after processing, we put at the beginning and at the end of the node sequence a node that computes the η-value (a measure of slowness, see Wiskott and Sejnowski, 2002) of its input (`EtaComputerNode`). The slow feature should match the driving force up to a scaling factor, a constant offset and the sign. To allow a direct comparison we rescale the driving force to have zero mean and unit variance. The real driving force is plotted together with the driving force estimated by SFA in Figure 6.6.

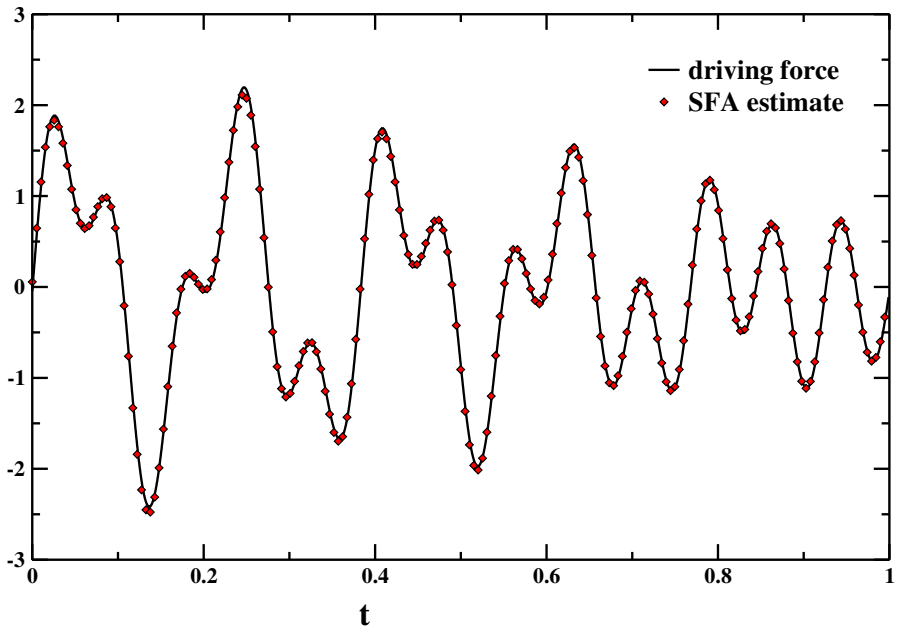


Figure 6.6: The real driving force and the driving force as estimated by SFA.

6.5 Future development

MDP is currently maintained by a core team of 3 developers, but it is open to user contributions. Users have already contributed some of the nodes and more contributions are currently being reviewed for inclusion in future releases of the package. The package development can be followed on the public subversion code repository[7]. Questions, bug reports, and feature requests are typically handled by the user mailing list[8].

Development of the core functionality of MDP continues and the next release of MDP is going to include a new package for parallelization, designed for nodes in which a large part of the computation is *embarrassingly parallel*[9] (e.g. calculating the covariance matrix to perform PCA). The new parallel package will consist of two parts: The first part introduces parallel versions of the familiar MDP structures (nodes and flows, including `hinet`) that are able to split the computations for some of the algorithms (e.g., PCA and SFA). The second part of the package consists of schedulers that take individual jobs and execute them in a parallel way. Currently a scheduler for parallelization across multiple processors (or cores) is provided. Since the scheduler code is largely independent of MDP,

[7] `http://mdp-toolkit.svn.sourceforge.net`

[8] `http://sourceforge.net/mail/?group_id=116959`

[9] In the jargon of parallel computing, an *embarrassingly parallel* problem is one for which no particular effort is needed to segment the problem into a very large number of parallel tasks, that can be executed more or less independently, without communication among tasks (Foster, 1995, sec. 1.4.4.).

one can write simple adapters for other schedulers like for example Parallel Python[10]. The new `parallel` subpackage can be tested already and it is available on the public code repository.

Another new, large MDP package is currently under development that will extend MDP with more complex data flows, including back-propagation and loops. This framework will be integrated with both the parallel and the `hinet` package to allow for large and complex data processing networks.

MDP could also act efficiently as a wrapper for the plethora of statistical data analysis algorithms already available in other libraries and languages. A prominent example is the R Project for Statistical Computing[11] with the Python wrappers RPy[12] and R/S Plus[13].

6.6 Conclusions

With over 20,000 downloads since its first public release in 2004, MDP has become one of Python's major scientific packages. The package has minimal dependencies, requiring only the NumPy numerical extension, is completely platform-independent, and is available in the Linux Debian distribution and the Python(x,y)[14] scientific Python distribution.

MDP has been used to implement a model of the visual system of a virtual rat moving around in a virtual environment (Franzius, Sprekeler, and Wiskott, 2007a), to perform pattern recognition (Franzius, Wilbert, and Wiskott, 2008a) and handwritten digit recognition (Berkes, 2006), to analyze intra-cerebral array-recorded neurophysiological data in the auditory forebrain of song birds[15], and to perform PCA and spike-sorting of electrophysiological data (Wiltschko, Gage, and Berke, 2008), to name a few of the applications in computational neuroscience. MDP has also been used embedded in the X-ray fluorescence mapping package PyMCA (Solé, Papillon, Cotte, Walter, and Susini, 2007), to implement auto tagging capabilities into the personal organizer application Chandler[16] by OSAF[17], and as a framework for the implementation of data processing algorithms in the context of an advanced course in scientific computing (Zito and Wilson, 2008) aimed at graduate students.

As the number of its users and contributors is increasing, MDP appears to be a good

[10] `http://www.parallelpython.com`
[11] `http://www.r-project.org/`
[12] `http://rpy.sourceforge.net/`
[13] `http://www.omegahat.org/RSPython/`
[14] `http://www.pythonxy.com`
[15] Gabriel J.L. Beckers, Max Planck Institute for Ornithology, Starnberg, Germany, *personal communication.*
[16] `http://chandlerproject.org/`
[17] `http://www.osafoundation.org/`

candidate for becoming a community-driven common repository of user-supplied, freely available, Python implemented data processing algorithms.

7 Conclusions

The notable achievements of models based on the slowness principle in the visual domain and the relative homogeneity of the cerebral cortex open the question of how well models based on the same principle would perform when applied to the auditory domain. This question has been addressed in several aspects in this thesis.

We have presented an algorithm for nonlinear blind source separation based on the slowness and statistical independence principles. While the algorithm performs well in many cases, a thorough analysis of the cases where it fails shows that the statistical independence principle, if implemented relying only on second order statistics, is not enough to ensure a satisfactory performance. Further work has been done in this direction: An algorithm solely based on the slowness principle and on a variational calculus approach that shows much better performance is under review for publication (Sprekeler, Zito, and Wiskott, 2010).

We have also shown the limitations of the slowness principle when it is applied to the auditory domain. The peculiar nature of the audio stimuli accounts for the results we obtained, which are very different from those known from the visual domain. Apart from some very simple cases, these results are not satisfactory. The reasons why the slowness principle shows such a poor performance are not completely understood. One approach may be to use a different representation in the pre-processing steps, or to introduce some data post-processing to help in the interpretation of the results. On the other hand, it may also be that the slowness principle is not general in the cortex, despite its noteworthy outcomes in the visual domain.

The main positive contribution of this work to the computational neuroscience community lies in the development and the release of the Modular toolkit for Data Processing, which is the basis of several scientific papers spanning diverse disciplines well beyond the domain for which it has been originally developed. The library is under constant development and aims at becoming a community-driven repository of well known and well tested machine learning algorithms.

Appendix A

Derivation of the ISFA Objective Function

The definitions of the constants d_n and e_n for the expression of the objective function (3.9) follow directly from the multilinearity of $C^{(\mathbf{u})}_{\cdots}(\tau)$. They are given in Table 1. Using trigonometry we can derive simpler objective functions of the form

$$\text{Case 1: } \Psi^{\mu\nu}_{\text{ISFA}}(\phi) = a_{20} + c_{24}\cos(4\phi) + s_{24}\sin(4\phi) \tag{1}$$

$$\begin{aligned}\text{Case 2: } \Psi^{\mu\nu}_{\text{ISFA}}(\phi) = a_{20} &+ c_{22}\cos(2\phi) + s_{22}\sin(2\phi) \\ &+ c_{24}\cos(4\phi) + s_{24}\sin(4\phi)\end{aligned} \tag{2}$$

with constants defined in Table 2. In the next step these objective functions are further simplified by combining the sine term and cosine term in a single cosine term. This results in:

$$\text{Case 1: } \Psi^{\mu\nu}_{\text{ISFA}}(\phi) = A_0 + A_4\cos(4\phi + \phi_4) \tag{3}$$

$$\text{Case 2: } \Psi^{\mu\nu}_{\text{ISFA}}(\phi) = A_0 + A_2\cos(2\phi + \phi_2) + A_4\cos(4\phi + \phi_4) \tag{4}$$

with constants defined in Table 3. It is easy to see why it is possible to write both objective functions (3) and (4) in such a simple form. Firstly, the terms in (3.9) are products of at most four $\sin(\phi)$ and $\cos(\phi)$ functions, which allows, at most, a frequency of 4. Secondly, in Case 1 $\Psi^{\mu\nu}_{\text{ISFA}}(\phi)$ has a periodicity of $\pi/2$ because rotations by multiples of $\pi/2$ correspond to a permutation (possibly plus sign change) of the two components. Since both components are inside the subspace, permutations do not change the objective function and the objective function has a $\pi/2$ periodicity. Thus we conclude that only frequencies of 0 and 4 can be present in (3). In Case 2, since one component lies outside the subspace, an exchange of components will change the objective function (4). A rotation by multiples of π, however, which results only in a possible sign change, will leave the objective function unchanged, resulting in an objective function with π-periodicity and therefore frequencies of 0, 2, and 4.

	Case 1	Case 2
d_0	$\sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \left(C^{(\mathbf{u}')}_{\mu\mu}\right)^2 + \left(C^{(\mathbf{u}')}_{\nu\nu}\right)^2$	$\sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \left(C^{(\mathbf{u}')}_{\mu\mu}\right)^2$
d_1	$4 \sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \left(C^{(\mathbf{u}')}_{\mu\mu} C^{(\mathbf{u}')}_{\mu\nu} - C^{(\mathbf{u}')}_{\mu\nu} C^{(\mathbf{u}')}_{\nu\nu}\right)$	$4 \sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} C^{(\mathbf{u}')}_{\mu\nu} C^{(\mathbf{u}')}_{\mu\mu}$
d_2	$2 \sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \left(2\left(C^{(\mathbf{u}')}_{\mu\nu}\right)^2 + C^{(\mathbf{u}')}_{\mu\mu} C^{(\mathbf{u}')}_{\nu\nu}\right)$	$2 \sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \left(2\left(C^{(\mathbf{u}')}_{\mu\nu}\right)^2 + C^{(\mathbf{u}')}_{\mu\mu} C^{(\mathbf{u}')}_{\nu\nu}\right)$
d_3	0	$4 \sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} C^{(\mathbf{u}')}_{\mu\nu} C^{(\mathbf{u}')}_{\nu\nu}$
d_4	0	$\sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \left(C^{(\mathbf{u}')}_{\nu\nu}\right)^2$
d_c	$\sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \sum_{\substack{\alpha=1,\\ \alpha\notin\{\mu,\nu\}}}^{R} \left(C^{(\mathbf{u}')}_{\alpha\alpha}\right)^2$	$\sum_{\tau\in T_{\text{SFA}}} \kappa^{\tau}_{\text{SFA}} \sum_{\substack{\alpha=1\\ \alpha\neq\mu}}^{R} \left(C^{(\mathbf{u}')}_{\alpha\alpha}\right)^2$

	Case 1	Case 2
e_0	$2 \sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \left(C^{(\mathbf{u}')}_{\mu\nu}\right)^2$	$2 \sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \sum_{\substack{\alpha=1\\ \alpha\neq\mu}}^{R} \left(C^{(\mathbf{u}')}_{\mu\alpha}\right)^2$
e_1	$4 \sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \left(C^{(\mathbf{u}')}_{\mu\nu} C^{(\mathbf{u}')}_{\nu\nu} - C^{(\mathbf{u}')}_{\mu\mu} C^{(\mathbf{u}')}_{\mu\nu}\right)$	$\sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \sum_{\substack{\alpha=1\\ \alpha\neq\mu}}^{R} C^{(\mathbf{u}')}_{\mu\alpha} C^{(\mathbf{u}')}_{\alpha\nu}$
e_2	$\sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \left(C^{(\mathbf{u}')}_{\mu\mu} - C^{(\mathbf{u}')}_{\nu\nu}\right)^2 - 2\left(C^{(\mathbf{u}')}_{\mu\nu}\right)^2$	$2 \sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \sum_{\substack{\alpha=1\\ \alpha\neq\mu}}^{R} \left(C^{(\mathbf{u}')}_{\alpha\nu}\right)^2$
e_c	$2 \sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \left(\sum_{\alpha=1}^{R-1} \sum_{\beta>\alpha}^{R} \left(C^{(\mathbf{u}')}_{\alpha\beta}\right)^2 - \left(C^{(\mathbf{u}')}_{\mu\nu}\right)^2\right)$	$2 \sum_{\tau\in T_{\text{ICA}}} \kappa^{\tau}_{\text{ICA}} \sum_{\substack{\alpha=1,\\ \alpha\neq\mu}}^{R-1} \sum_{\substack{\beta=\alpha+1,\\ \beta\neq\mu}}^{R} \left(C^{(\mathbf{u}')}_{\alpha\beta}\right)^2$

Table 1: Constants in Equation (3.9).

	Case 1	Case 2
a_{20}	$\frac{b_{\text{ICA}}}{4}(4e_c + e_2 + 3e_0)$ $-\frac{b_{\text{SFA}}}{4}(4d_c + d_2 + 3d_0)$	$\frac{b_{\text{ICA}}}{2}(2e_c + e_0 + e_2)$ $-\frac{b_{\text{SFA}}}{8}(8d_c + 3d_0 + d_2 + 3d_4)$
c_{22}	-	$\frac{b_{\text{ICA}}}{2}(e_0 - e_2) - \frac{b_{\text{SFA}}}{2}(d_0 - d_4)$
s_{22}	-	$\frac{b_{\text{ICA}}}{2}e_1 - \frac{b_{\text{SFA}}}{4}(d_1 + d_3)$
c_{24}	$\frac{b_{\text{ICA}}}{4}(e_0 - e_2) - \frac{b_{\text{SFA}}}{4}(d_0 - d_2)$	$-\frac{b_{\text{SFA}}}{8}(d_0 - d_2 + d_4)$
s_{24}	$\frac{b_{\text{ICA}}}{4}e_1 - \frac{b_{\text{SFA}}}{4}d_1$	$-\frac{b_{\text{SFA}}}{8}(d_1 - d_3)$

Table 2: Constants in Equation (1) and (2) in terms of the constants of Table 1.

	Case 1	Case 2
A_0	a_{20}	a_{20}
A_2	-	$\sqrt{c_{22}^2 + s_{22}^2}$
A_4	$\sqrt{c_{24}^2 + s_{24}^2}$	$\sqrt{c_{24}^2 + s_{24}^2}$
$\tan(\phi_2)$	-	$-\frac{s_{22}}{c_{22}}$
$\tan(\phi_4)$	$-\frac{s_{24}}{c_{24}}$	$-\frac{s_{24}}{c_{24}}$

Table 3: Constants in Equations (3) and (4) in terms of the constants of Table 2.

Bibliography

K. Achan, S. T. Roweis, and B. J. Frey. Probabilistic inference of speech signals from phaseless spectrograms. In *Neural Information Processing Systems 16 (NIPS'03)*, pages 1393–1400. MIT Press, 2003.

A. Aertsen and P. Johannesma. The spectro-temporal receptive field. a functional characteristic of auditory neurons. *Biol Cybern*, 42:133–143, 1981.

J. Allen. Nonlinear cochlear signal processing. In A. F. Jahn and J. Santos-Sacchi, editors, *Physiology of the Ear*, pages 393–442. Singular Thompson, 2001.

L. Almeida. Linear and nonlinear ICA based on mutual information - the MISEP method. *Signal Processing*, 84(2):231–245, 2004. Special Issue on Independent Component Analysis and Beyond.

S. Amari, A. Cichocki, and H. Yang. Recurrent neural networks for blind separation of sources. In *Proc. of the Int. Symposium on Nonlinear Theory and its Applications (NOLTA-95)*, pages 37–42, Las Vegas, USA, 1995.

M. Babaie-Zadeh, C. Jutten, and K. Nayebi. A geometric approach for separating post-nonlinear mixtures. In *Proc. of the XI European Signal Processing Conference (EUSIPCO 2002)*, pages 11–14, 2002.

A. Belouchrani, K. Abed Meraim, J.-F. Cardoso, and Éric Moulines. A blind source separation technique based on second order statistics. *IEEE Transactions on Signal Processing*, 45(2):434–444, 1997.

P. Berkes. Pattern recognition with slow feature analysis, February 2005. URL `http://cogprints.org/4104/`.

P. Berkes. *Temporal slowness as an unsupervised learning principle.* PhD thesis, Humboldt-Universität zu Berlin, Mathematisch-Naturwissenschaftliche Fakultät I, 2006. `http://edoc.hu-berlin.de/docviews/abstract.php?id=26704`.

P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex-cell properties. Cognitive Sciences EPrint Archive (CogPrints) 2804, `http://cogprints.ecs.soton.ac.uk/archive/00002804/`, Feb. 2003.

P. Berkes and L. Wiskott. Slow feature analysis yields a rich repertoire of complex cell properties. *Journal of Vision*, 5(6):579–602, July 2005. `http://journalofvision.org/5/6/9/`, doi:10.1167/5.6.9.

C. M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995.

C. M. Bishop. *Pattern recognition and machine learning*. Springer-Verlag, 2007.

T. Blaschke and L. Wiskott. CuBICA: Independent component analysis by simultaneous third- and fourth-order cumulant diagonalization. *IEEE Transactions on Signal Processing*, 52(5):1250–1256, 2004.

T. Blaschke, P. Berkes, and L. Wiskott. What is the relation between independent component analysis and slow feature analysis? *Neural Computation*, 18(10):2495–2508, Oct. 2006.

T. Blaschke, T. Zito, and L. Wiskott. Independent slow feature analysis and nonlinear blind source separation. *Neural Computation*, 19(4):994–1021, 2007.

E. Boer and P. Kuyper. Triggered correlation. *IEEE Trans Biomed Eng*, 15:169–179, 1968.

J. Cardoso. High-order contrasts for independent component analysis. *Neural Computation*, 11:157–192, 1999.

J.-F. Cardoso. The three easy routes to independent component analysis; contrasts and geometry. In *Proc. of the 3rd Int. Conference on Independent Component Analysis and Blind Source Separation, San Diego, (ICA 2001)*, 2001.

J.-F. Cardoso and A. Souloumiac. Blind beamforming for non Gaussian signals. *IEE Proceedings-F*, 140:362–370, 1993.

J.-F. Cardoso and A. Souloumiac. Jacobi angles for simultaneous diagonalization. *SIAM J. Mat. Anal. Appl.*, 17(1):161–164, 1996.

P. Comon. Independent component analysis, a new concept? *Signal Processing*, 36(3): 287–314, 1994. Special Issue on Higher-Order Statistics.

P. Comon and C. Jutten. *Handbook of Blind Source Separation*. Academic Press, Paris, 2010.

P. Comon, C. Jutten, and J. Herault. Blind separation of sources, Part ii : Problems statement. *Signal Processing*, 24:11–20, 1991.

T. Cover and J. Thomas. *Elements of information theory*. Wiley, New York, 1991.

S. Dähne, N. Wilbert, and L. Wiskott. Learning complex cell units from simulated prenatal retinal waves with slow feature analysis. In *Proc. 18th Annual Computational Neuroscience Meeting, CNS 2009, Berlin, July 18–23*, 2009. doi: 10.1186/1471-2202-10-S1-P129. URL `http://www.biomedcentral.com/1471-2202/10/S1/P129`. Special issue of BMC Neuroscience 10(Suppl 1):P129.

D. L. Donoho and C. Grimes. Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. *Proceedings of the National Academy of Sciences*, 100(10): 5591–5596, 2003.

M. Elhilali, T. Chi, and S. A. Shamma. A spectro-temporal modulation index (stmi) for assessment of speech intelligibility. *Speech Communication*, 41(2-3):331–348, Oct. 2003.

J. Ferwerda, S. Pattanaik, P. Shirley, and D. Greenberg. A model of visual adaptation for realistic image synthesis. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, pages 249–258, New York, US, 1996.

P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3 (2):194–200, 1991.

I. Foster. *Designing and building parallel programs.* Addison-Wesley, 1995.

M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, 2007a.

M. Franzius, H. Sprekeler, and L. Wiskott. Slowness and sparseness lead to place, head-direction, and spatial-view cells. *PLoS Computational Biology*, 3(8):e166, Aug. 2007b. (doi:10.1371/journal.pcbi.0030166).

M. Franzius, N. Wilbert, and L. Wiskott. Invariant object recognition with slow feature analysis. In *Proc. of the 18th International Conference on Artificial Neural Networks (ICANN'08)*. Springer Verlag, 2008a.

M. Franzius, N. Wilbert, and L. Wiskott. Invariant object recognition with slow feature analysis. In V. Kurková, R. Neruda, and J. Koutník, editors, *Proc. 18th Intl. Conf. on Artificial Neural Networks, ICANN'08, Prague*, volume 5163 of *Lecture Notes in Computer Science*, pages 961–970. Springer, Sept. 2008b. ISBN 978-3-540-87535-2. URL `http://dblp.uni-trier.de/db/conf/icann/icann2008-1.html#FranziusWW08`.

B. Fritzke. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, 1995.

D. Gabor. Acoustical quanta and the theory of hearing. *Nature*, 159:591–594, 1947.

T. D. Griffiths and J. D. Warren. What is an auditory object? *Nat Rev Neurosci*, 5(11): 887–892, 2004.

S. Harmeling, A. Ziehe, M. Kawanabe, and K.-R. Müller. Kernel-based nonlinear blind source separation. *Neural Computation*, 15:1089–1124, 2003.

H. Hartline. The receptive fields of optic nerve fibers. *Am J Physiol*, 130:690–699, 1940.

G. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18:1527–1554, 2006.

C. Hinze, N. Wilbert, and L. Wiskott. Visualization of higher-level receptive fields in a hierarchical model of the visual system. In *Proc. 18th Annual Computational Neuroscience Meeting, CNS 2009, Berlin, July 18–23*, 2009. doi: 10.1186/1471-2202-10-S1-P158. URL `http://www.biomedcentral.com/1471-2202/10/S1/P158`. Special issue of BMC Neuroscience 10(Suppl 1):P158, (abstract).

S. Hosseini and C. Jutten. On the separability of nonlinear mixtures of temporally correlated sources. *IEEE Signal Processing Letters*, 10(2):43–46, 2003.

A. Hyvärinen. Fast and robust fixed-point algorithms for independent component analysis. *IEEE Transactions on Neural Networks*, 10:626–634, 1999.

A. Hyvärinen and P. Pajunen. Nonlinear independent component analysis: existence and uniqueness results. *Neural Networks*, 12(3):429–439, 1999.

I. Jolliffe. *Principal component analysis*. Springer-Verlag, 1986.

C. Jutten and J. Herault. Détection de grandeurs primitives dans un message composite par une architecture de calcul neuromimétique en apprentissage non supervisé. In *Proc. of GRETSI'85*, pages 1017–1022, 1985.

C. Jutten and J. Karhunen. Advances in nonlinear blind source separation. In *Proc. of the 4th Int. Symposium on Independent Component Analysis and Blind Signal Separation, Nara, Japan, (ICA 2003)*, pages 245–256, 2003.

K. P. Körding, P. König, and D. J. Klein. Learning of sparse auditory receptive fields. In *Neural Networks. Proc. of the International Joint Conference on Neural Networs IJCNN 2002*, pages 1101–1108, 2002.

T.-W. Lee, M. Girolami, and T. Sejnowski. Independent component analysis using an extended Infomax algorithm for mixed sub-Gaussian and super-Gaussian sources. *Neural Computation*, 11(2):409–433, 1999.

S. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Trans. Acoust. Speech Sig. Proces.*, 37(12):2091–2110, 1989.

P. McCullagh. *Tensor methods in statistics*. Monographs on Statistics and Applied Probability. Chapmann and Hall, London, 1987.

G. Mitchison. Removing time variation with the anti-hebbian differential synapse. *Neural Computation*, 3(3):312–320, 1991.

L. Molgedey and G. Schuster. Separation of a mixture of independent signals using time delayed correlations. *Physical Review Letters*, 72(23):3634–3637, 1994.

A. W. Roe, S. L. Pallas, Y. H. Kwon, and M. Sur. Visual projections routed to the auditory pathway in ferrets: receptive fields of visual neurons in primary auditory cortex. *J. Neurosci.*, 12:3651–3664, Sep 1992.

S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326, 2000.

J. Sharma, A. Angelucci, and M. Sur. Induction of visual orientation modules in auditory cortex. *Nature*, 404:841–847, Apr 2000.

V. A. Solé, E. Papillon, M. Cotte, P. Walter, and J. Susini. A multiplatform code for the analysis of energy-dispersive x-ray fluorescence spectra. *Spectrochimica Acta Part B*, 62(1):63–68, 2007.

H. Sprekeler, C. Michaelis, and L. Wiskott. Slowness: An objective for spike-timing–dependent plasticity? *PLoS Computational Biology*, 3(6):e112, June 2007. (doi:10.1371/journal.pcbi.0030112).

H. Sprekeler, T. Zito, and L. Wiskott. An extension of slow feature analysis for nonlinear blind source separation, October 2010. URL `http://cogprints.org/7056/`.

J. Stone. Learning perceptually salient visual parameters using spatiotemporal smoothness constraints. *Neural Computation*, 8(7):1463–1492, 1996.

J. Stone and A. Bray. A learning rule for extracting spatio-temporal invariances. *Network*, 6(3):1–8, 1995.

M. Sur, A. Angelucci, and J. Sharma. Rewiring cortex: the role of patterned activity in development and plasticity of neocortical circuits. *J. Neurobiol.*, 41:33–43, Oct 1999.

A. Taleb. A generic framework for blind source separation in structured nonlinear models. *IEEE Transactions on Signal Processing*, 50(8):1819–1830, 2002.

A. Taleb and C. Jutten. Nonlinear source separation: The post-nonlinear mixtures. In *Proc. European Symposium on Artificial Neural Networks, Bruges, Belgium*, pages 279–284, 1997.

A. Taleb and C. Jutten. Source separation in post non linear mixtures. *IEEE Transactions on Signal Processing*, 47(10):2807–2820, October 1999.

F. E. Theunissen, K. Sen, and A. J. Doupe. Spectral-temporal receptive fields of nonlinear auditory neurons obtained using natural sounds. *The Journal of Neuroscience*, 20(6):2315–2331, 2000.

L. Tong, R. Liu, V. C. Soon, and Y.-F. Huang. Indeterminacy and identifiability of blind identification. *IEEE Transactions on Circuits and Systems*, 38(5):499–509, may 1991.

L. von Melchner, S. L. Pallas, and M. Sur. Visual behaviour mediated by retinal projections directed to the auditory pathway. *Nature*, 404:871–876, Apr 2000.

A. B. Wiltschko, G. J. Gage, and J. D. Berke. Wavelet filtering before spike detection preserves waveform shape and enhances single-unit discrimination. *Journal of Neuroscience Methods*, 173:34–40, 2008.

L. Wiskott. Learning Invariance Manifolds. In *Proc. of the 5th Joint Symp. on Neural Computation, May 16, San Diego, CA*, volume 8, pages 196–203, San Diego, CA, 1998. Univ. of California.

L. Wiskott. Estimating driving forces of nonstationary time series with slow feature analysis. arXiv.org e-Print archive, `http://arxiv.org/abs/cond-mat/0312317/`, 2003.

L. Wiskott and T. Sejnowski. Slow feature analysis: unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.

H.-H. Yang, S. Amari, and A. Cichocki. Information-theoretic approach to blind separation of sources in non-linear mixture. *Signal Processing*, 64(3):291–300, 1998.

X. Yang, K. Wang, and S. Shamma. Auditory representations of acoustic signals. *IEEE Trans. Inf. Theory*, 38(2):824–839, 1992.

D. Youla and H. Webb. Image restoration by the method of convex projections: Part 1-theory. *IEEE Trans. Med. Imaging*, MI-1(2):81–94, 1982.

A. Ziehe and K.-R. Müller. TDSEP – An efficient algorithm for blind separation using time structure. In *Proc. of the 8th Int. Conference on Artificial Neural Networks (ICANN'98)*, pages 675 – 680. Springer Verlag, 1998.

A. Ziehe, M. Kawanabe, S. Harmeling, and K.-R. Müller. Blind separation of post-nonlinear mixtures using linearizing transformations and temporal decorrelation. *Journal of Machine Learning Research*, 4:1319–1338, 2003.

T. Zito and G. Wilson. Software carpentry for scientists. `http://itb.biologie.hu-berlin.de/~zito/teaching/SC/`, 2008.

T. Zito, N. Wilbert, L. Wiskott, and P. Berkes. Modular toolkit for Data Processing (MDP): a Python data processing framework. *Frontiers in Neuroinformatics*, 2(8), 2008. (doi:10.3389/neuro.11.008.2008).

List of Figures

List of Tables

Selbständigkeitserklärung

Ich erkläre, dass ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Berlin, den 26.10.2010 Tiziano Zito